Öö pikkuse arvutamise rakendus

Karl Hendrik Holst

Käesolevas dokumentatsioonis on kirjeldatud rakenduse üldised tööpõhimõtted, disaini valikud, rakenduse kasutamine, lahendamisel tekkinud probleemid ja ajakulu. Ülesande püstitusest oli veidi keeruline aru saada, mida dokumentatsioonist täpselt oodatakse. Et kooliga ka jõuaks mingil määral tegeleda, otsustasin, et ei hakka dokumentatsioonis koodi ja funktsioone põhjalikult kirjeldama vaid annan üldise ülevaate sellest, mida ma tegin ja kuidas kood töötab.

Kasutusjuhend

1. Veendu, et arvutil on internetiühendus ja ava https://night-length-app.herokuapp.com/ veebibrauseris
2. Sisesta käsitsi huvipakkuva asukoha koordinaadid või kliki kaardil asukoha valimiseks.
3. Vali kuupäev, mille kohta infot soovid leida.
4. Vajuta nupule arvuta.
5. Soovi korral muuda kellaaega ja vaata, kuidas öö ja päev muutuvad kaardil valitud kuupäeva lõikes.

Tehniline disain

Kuna rakenduse juures on kõige olulisem selle lihtne kasutamine, programmeerisin tavalises veebibrauseris käivitatava rakenduse, kasutades Javascript'i, HTML'i ja CSS'i (sh Bootstrap'i). Kaardi kuvamiseks võtsin kasutusele Javascripti Leaflet.js teegi. Esialgselt ei olnud mul plaani kaasata serveripoolset koodi, kuid kuna öö pikkuse arvutamiseks oli vaja ka leida ajatsoone ja teisendada UTC ja kohaliku aja vahel, otsustasin kasutada node.js-i ning sellele koostatud lahendamiseks vajaminevaid pakette. Seevõrd muutus rakenduse internetti üleslaadimine veidi keerulisemaks ja oli vaja teenusepakkujat, kelle pilves saaks ka serveripoolset koodi jooksutada. Olen varasemalt kasutanud selleks Heroku't ja ka seekord laadisin rakenduse nende pilve. Rakendus ja kood on vormistatud inglisekeelsena.

Rakendusepuu näeb välja järgmine:

-client

    app.js

    sunrise-formula.js

    index.html

    style.css

index.js

Kaustas client on kõik failid, mis jooksevad kliendi enda seadmes. App.js sisaldab endas kliendipoolset loogikat, mille oluliseimad osad on DOM'i manipuleerimine vastavalt kasutaja enda sisendile ja Leafleti veebikaart ning selle kasutamise funktsioonid. Sunrise-formula.js sisaldab endas valemeid, et arvutada erinevaid päikesesündmuseid ja neid sobivale kujule muuta. Index.html on HTML document, mida kasutaja reaalselt näeb enda ekraanil, ehk seal on defineeritud kõik vajalik, et saada kasutajalt sisendandmed ning hiljem kuvada ekraanile väljundandmed. Style.css-is on defineeritud disain (lisaks kasutatud Bootstrap'i klassidele), mida HTML dokument kasutab. Index.js on serveripoolne koodifail, mis saadab vastuseid /timezone *endpointile*.

Öö pikkuse arvutamine

Öö pikkuse arvutamiseks oli vaja leida päikesetõusu ja päikeseloojangu (või mõne muu ööpäeva faasi) vaheline aeg. Kuna ülesande püstituses oli selgelt kirjas, et päikesetõus ja -loojang tuleb arvutada, matsin kohe maha mõtte kasutada mõne teenusepakkuja API't või mõnda node'i paketti vajalike andmete leidmiseks. Seetõttu hakkasin internetist otsima valemeid, millega leida päikesetõusu ja -loojangu kellaaeg, kasutades koordinaate ja kuupäeva. Esimesena ma üritasin rakendada NOAA valemit, mis hilisemal testimisel osutus liiga ebatäpseks (+- 20 minutit tegelikust sündmusest). Seejärel leidsin valemi, mis pärineb 1990 aastal välja antud raamatust *Almanac for Computers*. Antud valemi rakendamisel saadud tulemused olid täpsed ja seetõttu kasutab kood just seda valemit öö pikkuse arvutamiseks.

Eelmainitud valemi abil leiab rakendus päikesetõusu ja -loojangu ning tsiviil, nautilised ja astronoomilised ööpäeva faasid. Erinevuseks nende faaside vahel on nurk, mille all päike on sündmuse ajal. Päikeseloojangu ja -tõusu ajal on päike 90.85 kraadise nurga all seniidist (arvestades päikeseketta suurust ja keskmist refraktsiooni). Samad näitajad tsiviil, nautilisele ja astronoomilisele faasile on vastavalt 96, 102 ja 108 kraadi.

Algoritmi rakendamise tulemusena leitakse sündmuse UTC kellaaeg, mis seejärel on vaja teisendada kohalikku ajavööndisse. Selleks võtsin kasutusele kolm node paketti: geo-tz, mis leiab koordinaatide põhjal ajavööndi nimetuse, moment.js, mis leiab ajavööndi nimetuse põhjal selle erinevuse UTC kellaajast valitud kuupäeval ja express, mille abil on defineeritud *endpoint* andmete serveerimiseks. Lõpptulemuseks on päikesetõusu ja -loojangu kellaajad ning tsiviil, nautilise ja astronoomilise perioodi lõpukellaajad.

Koodis on sündmuste vahemike leidmisel loogiliselt eraldatud kolm olukorda: kui päike ei looju, kui päike ei tõuse ja kui päike loojub ning tõuseb. See võimaldas lihtsustada koodi kirjutamist, kuna esines olukordi, kus mõni videviku/koidu periood kestab üle kesköö või kus mõnel perioodil on kahe vahemiku asemel üks või kus mõni periood üldse puudub. Kõigi nende universaalne haldamine oleks olnud oluliselt keerulisem ja veaaltim.

Kuna öine aeg on võrdemisi subjektiivne mõiste, siis oli vaja ka välja mõelda, kuidas defineerida öö. Kõige lihtsamini mõistetav oleks päikeseloojangu ja -tõusu vaheline aeg. Kuid tegelikkuses ei peegelda see täpselt öist (kõige pimedamat) aega, kuna mõnda aega enne päikesetõusu ei ole pime ja ka peale päikeseloojangut ei lähe kohe pimedaks. Seetõttu otsustasin, et öine aeg jääb astronoomilise koidiku ja astronoomilise videviku vahele (ehk kui päike on horisondist madalamal kui 18 kraadi). Sama valemit kasutab ka timeanddate.com.

Veebikaart ja päikeseloojangu kujutamine

Veebikaardi kuvamiseks kasutab rakendus Leaflet.js teeki ja Carto taustakaarti. Öö ja päeva (päikeseloojangu ja -tõusu põhjal) eraldusjoone kuvamiseks kasutab rakendus Leaflet pluginat Terminator. Samasuguse tulemuse oleks võimalik olnud saavutada ka ise valemeid rakendades, kuid kuna seekord oli ülesandes öeldud, et kuva, mitte arvuta, otsustasin mitte minna ratta taasleiutamise teed ja kasutada kõige lihtsamat ning töötavat lahendust. Mainitud plugin võtab parameetrina sisse Javascript'i kuupäeva objekti ja kuvab polügonina ala, kus päike on antud ajahetkel loojunud. Lisaks on rakenduse kasutajal võimalik interaktiivselt terminaatorit kuvada eri kuupäevadel ja kellaaegadel, vaikimisi on selle väärtuseks praegune kohalik kellaaeg.

Kliendi poolne sisend

Kasutajalt vajalike andmete saamiseks on HTML dokumendis defineeritud kolm sisendvälja, milleks on pikkuskraad, laiuskraad ja kuupäev. Kasutades *eventlistener'*e toimub pidev sisendikontroll (laiuskraad peab jääma vahemikku -90 ja 90, pikkuskraad vahemikku -180, 180, koordinaadid peavad olema numbrilised väärtused, kuupäev ei tohi olla väärtuseta). Vajadusel kuvatakse kasutajale ka veateade.

Lisaks käsitsi koordinaatide sisestamisele, on võimalik klikkida kaardil koordinaatide valimiseks, mis kasutab Leafleti sisseehitatud funktiooni, mille abil leitakse valitud asukoha koordinaadid ja täidetakse automaatselt koordinaatide sisendväljad. Valitud koordinaate tähistab kaardil asukohamarker. Kui kõik vastavad väljad on valideeritud, peab klient vajutama arvuta nupule, mis jooksutab vajalikku Javascripti koodi, et leida öö pikkus, päikesetõusu ja -loojangu kellaeg ja ööpäeva eri faasid (kellaajaliselt). Seejärel kirjutatakse üle vastavate DOM elementide sisu ja arvutatud tulemused kuvatakse kaardi alla.

Samuti on kaardi all *range* sisendelement, mille võimalikud väärtused jäävad vahemikku 0 ja 1440 (minuteid ööpäevas) ning mille sammu pikkus on 1 minut. Ajaliste sisendandmete muutumisel luuakse uus kuupäeva objekt vastavalt valitud kellaajale ja kuupäevale, mis seejärel antakse argumendina öö ja päeva terminaatori *setTime()* funktsiooni, mille tulemusena joonistatakse terminaator uuesti.

Serveripoolne kood

Nagu eelnevalt mainitud, on ajatsooni ja UTC *offseti* leidmiseks valitud koordinaatidel kasutatud kahte node.js-i paketti. Lisaks neile on Express'i paketiga defineeritud /timezone *endpoint*, mis vastab kliendipoolsest koodist saadetud GET *request'ile*. Asukoha koordinaadid ja kuupäev edastatakse GET *request'i* parameetritena URLis. Kasutades moment.js'i ja geo-tz'i, leiab kood kohaliku kellaaja erinevuse UTC kellaajast, mis seejärel saadetakse json formaadis vastusena päringule. Leitud väärtust on seejärel kasutatud öö pikkuse ja muude lisaandmete leidmiseks.

Lahendamisel tekkinud probleemid

Minu jaoks oli kahtlemata kõige keerulisem osa päikesetõusu valemi rakendamine, kuna see koosnes paljudest osadest ja seda oli võrdlemisi keeruline siluda. Kui lõpplahendus oli vale,

siis ei olnud võimalik täpselt kindlaks teha, kus mingi viga on, vaid pidi algusest peale näpuga järge ajama. Samuti vajas põhjalikumat nuputamist kraadide ja radiaanide kasutamine, kuna Javascript kasutab trigonomeetrilistes valemites radiaane, aga valemis olid nurgad antud kraadides. Seega ma pidin valemi rakendamisse panustama oluliselt rohkem aega kui olin planeerinud. Kaasa ei aidanud ka esimesena katsetatud NOAA valem, mille ma suutsin õigesti implementeerida, aga hiljem lisainfot otsides, leidsin, et antud valem ongi liiga üldine ja ei anna soovitult täpseid tulemusi.

Samuti vajas põhjalikku loogilist mõtlemist erinevate ööpäeva perioodide vahemike leidmine nende lõpukellaegade põhjal. Alguses ma üritasin geneeriliselt neid vahemikke kokku saada, aga mingi hetk kasvas loogika üle pea ja seejärel otsustasin, et käsitlen kolme eelnevalt mainitud eri juhtu eraldi *if* tingimuste sees. Tulemusena on nüüd veidi rohkem koodi, kuid seda on lihtsam hallata ja sellest on lihtsam aru saada.

Üleüldiselt mujal ma enam väga hätta ei sattunudki, isegi kui olid mingi probleemid, andis kiire googeldamine vastuseid, kuidas neid lahendada. Peamised allikad, mida ma probleemide lahendamiseks kasutasin olid:

- StackOverflow
- MDN'i dokumentatsioon
- Leaflet'i dokumentatsioon
- Node.js ja kasutatud pakettide dokumentatsioon
- Bootstrap'i dokumentatsioon

Ajakulu

Järgnevalt toon välja erinevate rakenduse osade ajakulu:

- Päikesetõusu valemite implementeerimine: 25 tundi
- Valemite tulemuste inimkeelde tõlkimine: 8 tundi
- Öö ja päeva kujutamine kaardil: 2 tundi
- Kliendipoolne Javascripti kood: 12 tundi
- Serveripoolne Javascripti kood: 1 tund
- HTML dokumendi koostamine: 3 tundi
- CSS: 1 tund
- Dokumentatsioon: 4 tundi

Kokku: 55 tundi