Dokumentasjon

IT 2 - Eksamen

Kandidatnummer: 569HKG-V

Fagkode: REA3015JAV

29.05.17

Innholdsfortegnelse

Innledning

Litteraturliste

- It-2 boka
- «JVisuals Library» - *Kandidat* - 01.05.17

Programvareoversikt

	Programvare	Versjonnummer
Operativsystem	macOS Sierra	12.12.1 Beta
Kodeeditor	Atom	10.15.0
Kjøremiljø/ Nettleser	Google Chrome	Versjon 53
Lydredigering	Adobe Premiere	10.0.4
Bilderedigering	Adobe Photoshop	2015.5.1
Videoredigering	Adobe Premiere	10.0.4
Tekstredigering	Pages	6.1.1
Design	Keynote	7.1

Utviklingsprosessen

Underveis i IT 2 har jeg fulgt en utviklingsprosess gitt ved spesielle steg. Det er denne utviklingsprosessen jeg vil ta utgangspunkt når jeg løser oppgavene i eksamenssettet.

-Krav

I kravprosessen kartlegges hva som skal produseres i utviklingsprosessen. Det blir ofte gitt en beskrivelse av kravene ved hjelp av en kravspesifikasjon. Denne kan

for eksempel komme fra en kunde. I en eksamenssituasjon er blir kravene gitt i eksamensoppgaven.

-Design

I designprosessen blir det lagt til grunn hvordan en skal gå frem for å løse oppgaven. Her blir det ofte laget en skisse over ulike designelementer som bør forekomme i oppgaven (wireframes). Det er også vanlig å bygge strukturen for utviklingsmiljøet her, hvilke programmeringsspråk en skal bruke, og hvilke program en skal bruke. En kan også legge til pseudokode for å gi en forsmak på hvordan programmeringsløsningene skal være.

-Kode

Det er i denne fasen at løsningen produseres. Her implementeres designet ved hjelp av programmeringsspråkene og programvaren en har valgt. Det er for øvrig sjeldent at prosessene design og kode skjer hver for seg. Et mer realistisk scenario er den at første desginfasen fungerer som en grovskrisse. Etter å ha kodet denne løsningen kan en gå tilbake til designfasen med hensikt om å forbedre den.

-Test

I testfasen skal koden/produktet testes ut. Her er det viktig å dokumentere ulike feil som oppstår og hvilke måter en kan forbedre koden/produktet. Dersom testfasen gir et positivt resultat går en videre til produksjonssetting.

-Produksjon

I denne fasen lanseres koden/produktet til kunden. Dette blir produksjonssatt dersom kunden sier seg fornøyd med utgangspunkt i kravene som ble stilt innledningsvis. Det er viktig å legge til en logg, eller et såkalt versjonsnotat for å holde oversikt over de ulike versjonene.

Mappestruktur

I eksamensbesvarelsen har jeg valgt å organisere filene på denne måten:

- Biblotek

Under Biblotek ligger filene fra biblotekene jeg har brukt for å løse eksamensoppgaven.

- Dokumentasjon

Under Dokumentasjon ligger dette dokumentet, hvor beskrivelse blant annet beskrivelse av utviklingsprosess og beskrivelse av løsningen til de ulike oppgavene ligger. I tillegg finner en dokumentasjon for biblotekene som blir brukt i besvarelsen.

- Oppgave #

Under Oppgave # er løsningene av de ulike oppgavene. Jeg har valgt å strukturere hver oppgave i en egen mappe fordi jeg ikke ønsker plutselig konflikt mellom ulike oppgaver i en eksamenssituasjon. I en optimal utviklingsprosess ville denne metoden blitt brukt i utviklingsprosessen, mens det ferdige produktet, her besvarelsen, ville blitt samlet i én løsning.

Struktur for besvarelse

Hver Oppgave # vil i utgangspunktet være strukturert på følgende måte:

- index.html

Her ligger kun html-elementene knyttet til den spesifikke siden/applikasjonen. Denne filen kan ha forskjellige navn avhengig av oppgaven og hvor mange html-filer en trenger.

- script.js

script.js er javascriptkoden spesifikt knyttet til index.html / et gitt html-dokument. Denne filen kan også ha forskjellige navn avhengig av oppgaven. Et

eksempel kan være at en har en fil produkt.html, en javascript-fil knyttet til denne html-filen vil naturligvis være produkt.js.

- methods.js

I methods.js finner en alle de generelle funksjonene/metodene som blir brukt på tvers av javascriptfilene.

- style.css

I style.css ligger css-koden for alle html-filene.

- .json

Dersom en har behov for en .json-fil vil også denne ligge i Oppgave #. .json-filen vil få et passende navn basert på hva den inneholder. Eksempelvis produkter.json.

- Media

Under Media ligger all multimedia som blir brukt i oppgaven.

Jeg har valgt å strukture Oppgave # på denne måten fordi det gir god oversikt og legger grunnlag for en rask utviklingsprosess. Istedenfor å ha forskjellige språk og kode med ulik tilhørighet i en fil er det på denne måten lett å navigere seg i koden og mellom de ulike filene, slik at feilsøking og programmering går raskere.

Besvarelse

Jeg vil i hver oppgave si noe om utviklingsprosessen.

Oppgave 1

-Krav

Jeg forstår kravene slik at animasjonen primært skal bestå av tre ting: en roterende vinmølle, beveglige landskapsobjekter, og et statisk landskap. Jeg har dermed valgt å benytte med av et biblotek som heter JVisuals. Med dette bibloteket kan jeg lage og manipulere objekter i en canvas på en relativt lett måte. Jeg ser for meg tre knapper under canvasen som styrer vindstyrken, oppdaterer canvasen og oppdaterer vindinformasjonen.

-Design

I og med at oppgaven ga mulighet for å «utforme bildet helt annerledes» valgte jeg å fokusere enkelthet istedenfor kompleksitet i canvasen. På denne måten vil tiden bli bedre til å lage god kode. Jeg ser for meg at bildet skal inneholde et tre med beveglige blad hvor hvor mye de beveger seg er avhengig av vindstyrken, og en vindmølle med ett roterende blad.

-Kode

Jeg valgte som sagt å bruke biblotektet JVisuals for å manipulere canvasen. På grunn av dette er hvert element i bildet definert som et objekt. Via ulike funksjoner beveger jeg objektene basert på vindstyrken.

-Produksjon

Løsningen har oppnådd kravene som ble satt innledningsvis. Dersom jeg skulle jobbet videre med dette prosjektet ville jeg fokusert på et penere bilde og å ha generalisert funksjonene enda mer, spesielt funksjonen som beveger bladene (denne ville jeg hatt inn i JVisuals.). Alt ser for øvrig ut til å funke slik jeg så for det for meg i starten.

Oppgave 2 / 3

-Krav

Jeg tolker det slik at Oppgave 2 og Oppgave 3 henger sammen og at jeg derfor kan si noe om kravene til begge samtidig. I begge oppgavene er det en rutine / algoritme som skal lages for å behandle ulike tallverdier. Tallverdiene skal så sees ut ifra en tabell med vindstyrker. Det er derfor viktig å lage generelle funksjoner som er uavhengig av bruksområde.

-Design

I begge oppgavene har jeg gått for et enkelt design hvor hovedfokuset er at funksjonene skal settes i en intuitiv sammenheng.

Oppgave 3

3A

Pseudokode for Oppgave 3:

For å kunne hente ut informasjon basert på hvilken vindstyrke brukeren skriver inn og bruke denne informasjonen til gjøre om til effekt og deretter legge sammen summen av all effekten, trenger vi hovedsakelig tre funksjoner.

Den første funksjonen kan vi kalle convert og har et parameter input som er et desimaltall. I convert sammenlignes dette tallet med verdier fra en liste ved hjelp av en for-løkke. I listen er verdiene gitt trinnvis i ulike mengder. Feks [0,2], hvor 0 er den laveste og 2 er den høyeste verdien. Ut i fra hvilket trinn tallet input ligger i kan vi finne vindtype og effekt. Siden minste vindtype som gir effekt er 2.5m/s og høyeste er 15m/s, men vindtypene er fortsatt definert utifra de områdene, kan vi legge til en if-statement som sjekker om input innenfor definisjonen. Convert returnerer så en array med [effekt, vindtype] for den gitte vindstyrken.

Den andre funksjonen vi trenger kaller jeg summarize. Summarize tar inn en parameter array med tallverdier. I en for-løkke adderes alle tallene i array til en sum, og funksjoen returnerer så denne summen.

Den siste funksjoen vi trenger kaller jeg displayWind. I displayWind hentes det ut informasjon fra ulike input-felt som representerer vindstyrke. Med convert får vi returnert effekten og vindtypen for vindstyrken, deretter må dette multipliseres med seks (pga tidsperioder) og til slutt summeres de ulike effektene for hver tidsperiode i summarize. Displaywind viser de ulike verdiene i html-elementer.

3B

Dokumentasjon av koden er lagt til som kommentar i selve koden.