HTML og GitHub

RASMUS JUEL NIELSEN

Datatekniker med speciale i Programmering

Indholdsfortegnelse

Indledning

I opgaven skal der udarbejdes en personlig portfolio hjemmeside ved brug af HTML. I opgaven benyttes der CSS og JavaScript til design af hjemmesidens komponenter - koden udarbejdes i Visual Studio Code. Der skal oprettes en bruger på GitHub, samt et repository for hjemmesiden, som derefter skal klones til VS Code så der kan foretaget ændringer direkte i VS Code som så overføres til GitHub via commits. Derudover skal der demontreres en række kommandoer i Git Bash for at vise hvordan det kan benyttes, samt forklares en række begreber omhandlende Git og hvordan de fungerer i arbejdsstrukturen.

Link til min hjemmeside: https://rasmusjueln.github.io

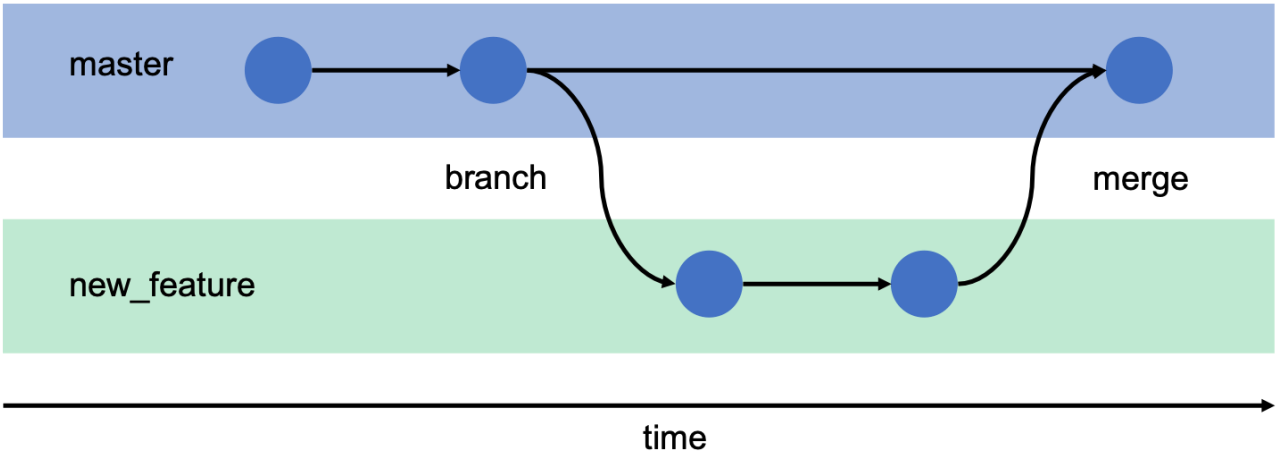
Hvad er Git

Når man arbejder på et projekt, enten alene eller sammen med andre, og der er en eller flere der laver ændringer i den samme kode, og måske på samme tid, så er Git et fantastisk værktøj til at bevare overblikket.

Git fungerer overordnet som et open source system til versionsstyring, hvor versionerne er de ændringer der bliver implementeret af de brugere der arbejder på projektet. Disse versioner eller ændringer bliver lagret i noget som kaldes repositories, og som derefter kan gennemgås inden de bliver endeligt commited til den eksisterende kode.

At Committe en ændring til en kode betyder at man tilføjer et nyt stykke kode til et eksisterende projekt, så en ny version af projektet oprettes. Der kan til hvert commit skrives en linje eller stikord om hvad ændringerne indebærer, så man hurtigt kan skabe sig et overblik hvor de forskellige ændringer befinder sig, hvis man senere vil revurdere den specifikke version af projektet.

Git råder over både lokal og webbaseret projekthåndering i form af henholdsvis Git Bash (lokalt) og GitHub (web). Via GitHub kan man ved brug af funktionen ”pages” udgive sin egen live hjemmeside og give det ønskede domæne navn. Pages læser den lokale index.html fil i repositoriet, det er derfor vigtigt at denne ligger i root folderen. Root folderen er den repositoriet læser først, som kan indeholde yderligere filer og mapper med kode som repositoriet kan bruge.



Denne figur viser eksempel på hvordan et projekt kan bygges op. Master er standard grenen/banchen, og er der hvor det primære projekt ligger. I forbindelse med at der skal ændres på noget kode oprettes der en ny branch. En branch gør det muligt at lave en ny version af koden med ændringer uden at det ændrer på den primære kode. Dette er vigtigt i forhold til at beholde den fungerende fil, så der ikke ved en fejltagelse bliver slettet vigtige komponenter eller at koden ikke kan køres efter ændringer – et slags sikkerhedsnet. Efter at man har foretaget ændringer i den nye branch, kan de merges med den primære kode. Der kan sagtens være flere branches af en kode i gang på samme tidspunkt. Dette kan være smart hvis der sidder flere og arbejder på forskellige dele af en kode.

Oprettelse af projekt med Git Bash

```
rasnie@1Sal-13 MINGW64 ~
$ git config --global user.name "RasmusJueln"

rasnie@1Sal-13 MINGW64 ~
$ git config --global user.email "rasmusn686@hotmail.com"

rasnie@1Sal-13 MINGW64 ~
$ git config --global core.autocrlf true

rasnie@1Sal-13 MINGW64 ~
$ git init
Initialized empty Git repository in C:/Users/rasnie/.git/

rasnie@1Sal-13 MINGW64 ~ (master)
$ mkdir ~/workdir

rasnie@1Sal-13 MINGW64 ~ (master)
$ cd ~/workdir
```

```
rasnie@1Sal-13 MINGW64 ~/workdir (master)
$ git init
Initialized empty Git repository in C:/Users/rasnie/workdir/.git/

rasnie@1Sal-13 MINGW64 ~/workdir (master)
$ git status
On branch master

No commits yet

nothing to commit (create/copy files and use "git add" to track)
```

```
rasnie@1Sal-13 MINGW64 ~/workdir (master)
$ touch Readme.txt

rasnie@1Sal-13 MINGW64 ~/workdir (master)
$ git add Readme.txt

rasnie@1Sal-13 MINGW64 ~/workdir (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   Readme.txt


rasnie@1Sal-13 MINGW64 ~/workdir (master)
$ git commit -m "added Readme.txt to repo"
[master df310a0] added Readme.txt to repo
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 Readme.txt

rasnie@1Sal-13 MINGW64 ~/workdir (master)
$ git log
commit df310a076518e44b9eb88d958e9e28093232fc94 (HEAD -> master)
Author: RasmusJueln <rasmusn686@hotmail.com>
Date:   Mon Feb 7 08:39:58 2022 +0100

    added Readme.txt to repo
```

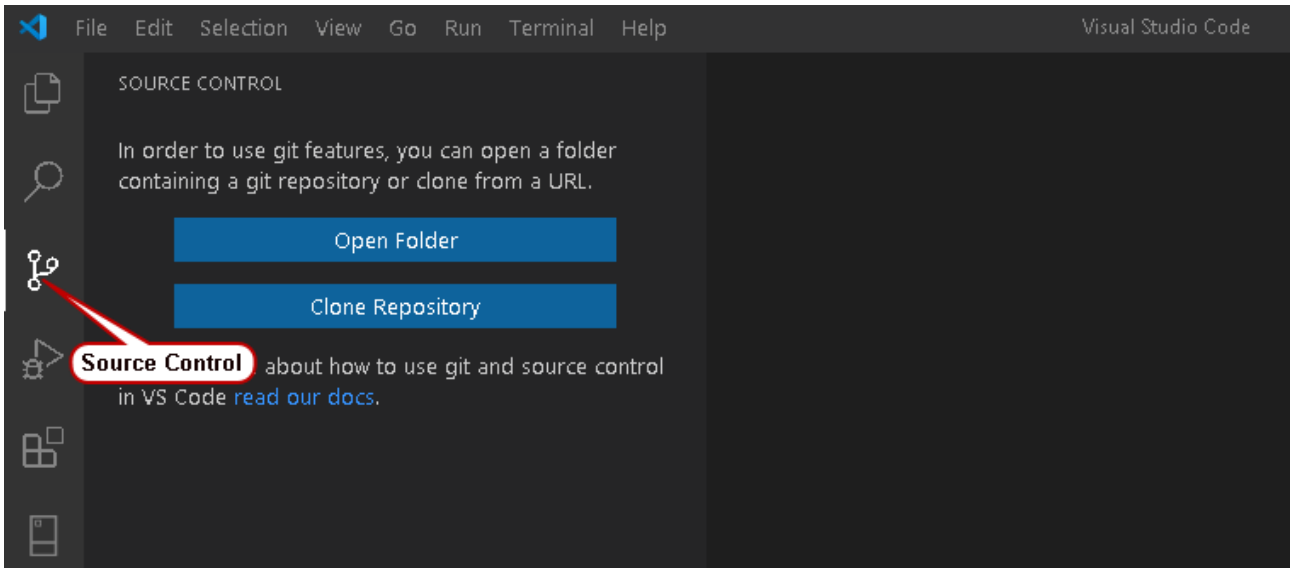
```
rasnie@1Sal-13 MINGW64 ~/workdir (master)
$ echo hello >> Readme.txt

rasnie@1Sal-13 MINGW64 ~/workdir (master)
$ git add Readme.txt
warning: LF will be replaced by CRLF in Readme.txt.
The file will have its original line endings in your working directory

rasnie@1Sal-13 MINGW64 ~/workdir (master)
$ git commit -m "added hello to Readme.txt"
[master 0398bc9] added hello to Readme.txt
 1 file changed, 1 insertion(+)

rasnie@1Sal-13 MINGW64 ~/workdir (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        deleted:    foo.txt

no changes added to commit (use "git add" and/or "git commit -a")

rasnie@1Sal-13 MINGW64 ~/workdir (master)
$ git log
commit 0398bc9ec4adb85d17cc4f2946ad08c6f647e239 (HEAD -> master)
Author: RasmusJueln <rasmusn686@hotmail.com>
Date:   Mon Feb 7 08:42:33 2022 +0100

    added hello to Readme.txt
```

Merge conflict

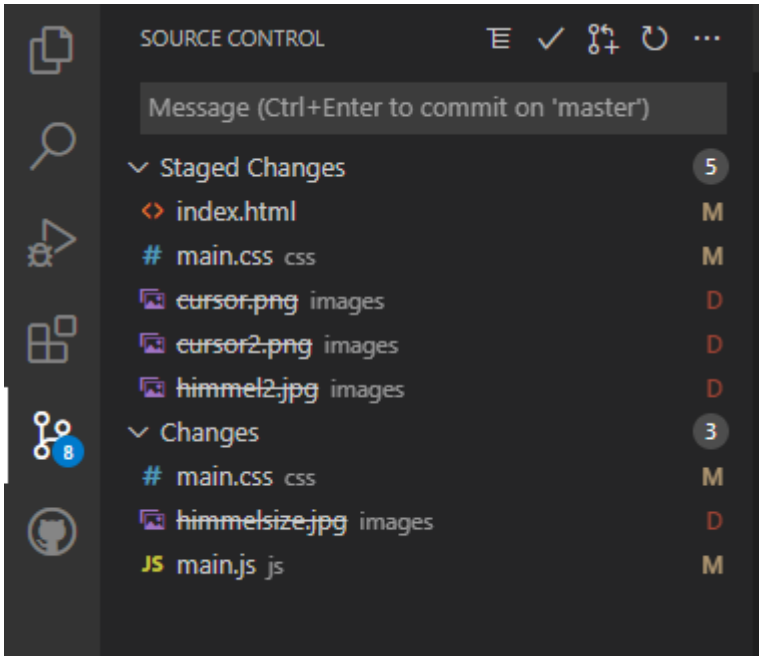
Hvis der sidder flere udviklere og arbejder på et projekt og en af disse udviklere forsørger at redigere en kode som en anden udvikler allerede redigerer kan der opstå det der kaldes merge conflict. For at undgå disse konflikter arbejder udviklerne i separate branches som nævnt tidligere. Her kommer Git merge kommandoens primære ansvar i brug da den går ind og kombinerer forskellige grene og løser de eventuelle konflikter der sker i forbindelse med modstridende redigeringer.

GitHub og VS Code

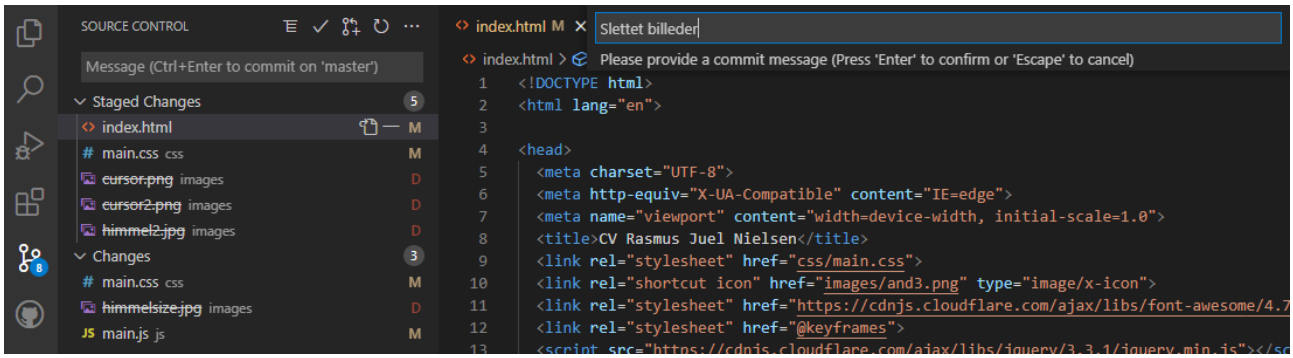
Jeg har oprettet en bruger på GitHub og oprettet et repository som hedder rasmusjueln.github.io, hvilket svarer til den kommende hjemmesides navn. I en mappe på C: drevet oprettes en mappe kaldet CV, hvori alle nødvendige materialer skal ligge til opbygning af hemmesiden, heriblandt css, javascript, billeder, dokumenter og selvfølgelig index.html filen. CV mappen uploades derefter til mit repository. Nu kan der laves en kobling fra GitHub til VS Code igennem Clone Repository funktionen under source control (se nedenstående billede).



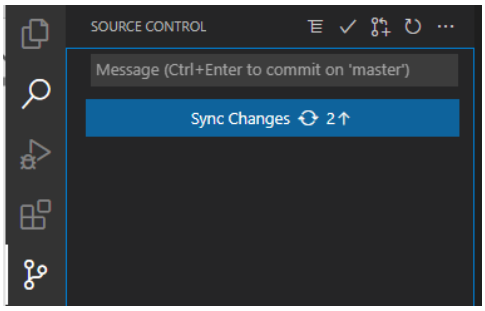
Efter repository og VS Code er klonet kan man nu lave ændringer i koden og derefter committe dem til repository, så de automatisk ændrer sig i koden på GitHub. Man kan også vælge at arbejde i en anden branch i repository, så når man laver et commit så ændrer den ikke på den primære kode i master branchen. Inden man committer kan man vælge hvilke hvilke ændringer der skal med i det commit. Det gøres ved at tilføje de valgte ændringer til stage changes.



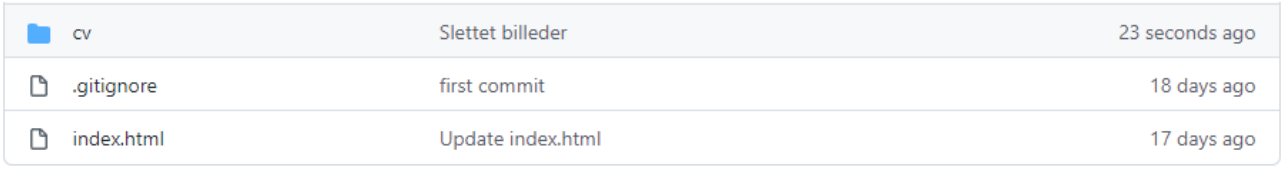
Derefter vælges commit (✓ tegnet i toppen af vinduet), og der tilføjes en kommentar til committet, som beskriver hvad ændringerne indeholder.



Derefter vælges Sync changes, som tilføjer ændringerne til GitHub repo.



Inde på GitHub kan man nu se at der er foretaget en ændring i mappen CV, hvornår de er foretaget og navnet på committet.



Konklusion

Overordnet har min tilgang til opgaven været lidt omvendt af hvad der nok var meningen, da jeg startede med at lave det meste af hjemmesiden inden jeg fik opsat min GitHub bruger og fik clonet med VS Code. Men jeg fik opsat min GitHub og manglede stadig noget på hjemmesiden, så jeg fik lært at bruge commits osv. efterhånden som jeg blev færdig med koden.

Jeg havde også en række udfordringer:

Efter jeg oprettede hjemmesiden som Pages og havde linket den til GitHub, kunne hjemmesiden ikke indlæses. Blev derfor nødt til at oprette ekstra index.html fil liggende i root folderen som ”linker” til mappen hvori den ønskede index.html fil ligger, så den ved hvilken fil der skal indlæses når hjemmesiden indlæses af browseren. Det blev løst med følgende fil kode:

```
1  <!DOCTYPE html>
2  <head><meta http-equiv="refresh" content="0; url=https://rasmusjueln.github.io/cv/index.html"></head>
3
4  </html>
```

Jeg havde store problemer med at linke min HTML med JavaScript, hvis jeg ikke skrev JavaScript koden direkte i HTML koden, ved brug af <script>. Dette skyldes at jeg ikke havde ”linket” til JavaScript koden i ”main.js” udenfor <body> under al andet kode. Jeg er stadig lidt i tvivl om hvorfor det ikke kunne stå i body delen.

```
<script src="js/main.js"></script>
```

Jeg har på min hjemmeside lavet en contact form, som jeg endnu ikke har fået til at fungere optimalt så den kan sende selve beskeden til min egen mail. Men det skyldes at jeg ikke har en live server som kan læse den php kode jeg har skrevet, og derfor ikke ved hvor den skal lagre informationen der bliver indtastet.