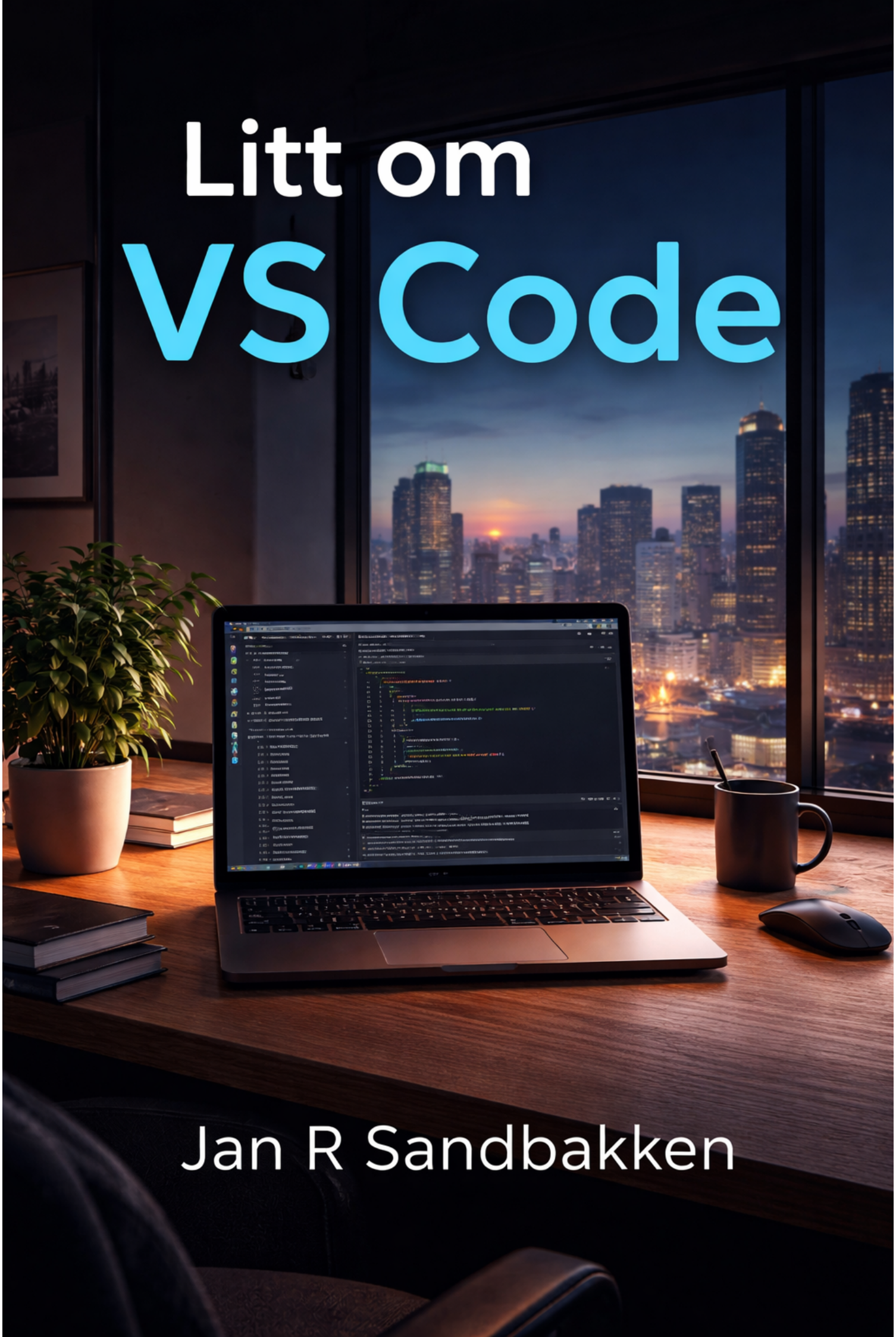
Litt om VS Code



Jan R Sandbakken

Litt om VS Code

Jan R Sandbakken

Version 1.0 2026-02-12

Innholdsfortegnelse

Litt om VS Code

Visual Code Studio (gjerne forkortet VS Code) er en gratis, lett og svært populær kildekode-editor utviklet av Microsoft. Den er designet for å hjelpe utviklere med å skrive, feilsøke og redigere kode for en rekke programmeringsspråk, og fungerer på Windows, macOS og Linux.

VS Code er/har:

- Lett og rask: Starter raskt og bruker lite systemressurser.
- IntelliSense: Tilbyr intelligent kodefullføring basert på variabeltyper, funksjonsdefinisjoner og importerte moduler.
- Innebygd Git-støtte: Gjør det enkelt å håndtere versjonskontroll direkte i editoren.
- Extensions (utvidelser): Man kan tilpasse editoren med tusenvis av utvidelser for nesten alle programmeringsspråk (Python, Java, PHP, Go, C++, etc.), for filformater som HTML, CSS, TypeScript, JSON, Markdown, Asciidoc etc, samt for temaer og ulike verktøy.
- Integrert terminal: Man kan kjøre kommandolinjeoppgaver uten å forlate editoren.
- Debugging: Innebygd støtte for å feilsøke kode, sjekke variabler og se *call stacks*.

Men med all denne funksjonaliteten og fleksibiliteten, følger også kompleksitet i oppsett og innstillinger. JSON er valgt for håndtering av innstillinger, og i starten kan den totale mengden virke overveldende. Likeledes kreves en rekke extensions for å jobbe med ulike programmeringsspråk eller filformater. Disse må oppdateres, stadig nye tilbys, og over tid kan oversikt og kontroll avta.

Dette heftet forsøker å være til hjelp for dem som ønsker å få en viss oversikt over VS Code. Det fokuseres altså på

- JSON-instillinger
- Utvidelser

VS Code ellers mange muligheter for å effektivisere arbeid. Det er bare å kaste seg over mulighetene. En av de vi kort skal se på er

- Task automation

Vi vil gjennomgående basere oss på et Linux-system her, slik at kataloger og eksempler i heftet gjelder Linux.

JSON

JSON (JavaScript Object Notation) er et tekstbasert format for lagring og utveksling av strukturerte data, hovedsakelig brukt mellom en server og en web-applikasjon. Det er uavhengig av programmeringsspråk og enkelt for maskiner å tolke. Dataene struktureres primært i nøkkel/verdi-par og lister (arrays). Datatyper inkluderer strenger, tall, boolske verdier, null, arrays og andre objekter.

Her ser vi eksempel på syntaks:

```
{
  "navn": "Ola Nordmann",
  "alder": 30,
  "erStudent": false,
  "kurs": ["Programmering", "Web-utvikling"]
}
```

JSON-innstillinger

JSON-innstillingene i VS Code er inndelt i følgende hierarki:

1. Folder settings (multi root settings)
2. Workspace settings (for hvert prosjekt)
3. User settings (globalt for brukeren)
4. Default settings (innebygd i VS Code)

De først nevnte vil overstyre de senere. Altså, om Workspace sier noe om `fontSize": 14`, mens User sier `fontSize": 12`, blir fontstørrelse 14 gjeldende.

Den første, multi root settings, er aktuell bare når VS Code har flere toppmapper åpne samtidig. Dette er for mer spesiell bruk, og vi skal ikke se særlig på det her. Det normale er å åpne VS Code på arbeidskatalogen til prosjektet, som gjerne må ha mange underkataloger. (Dette er likevel ikke multi root. Multi root startes ved å klikke **File/Add folder to Workspace ...** i VS Code-menyen og leder til en egen **.code-workspace**-fil.)

VS Code har i tillegg noen preferanser brukeren kan sette (ofte knyttet til layout som mørkt eller lyst tema, om paneler skal vises til høyre eller venstre etc) som ikke er en del av JSON-systemet og som lagres separat.

Workspace settings ligger i følgende fil på arbeidskatalogen:

```
<prosjektmappe>/.vscode/settings.json
```

User settings ligger i følgende fil:

```
~/.config/Code/User/settings.json
```

Det anbefales ikke å editere disse direkte, men heller bruke et GUI som VS Code.

JSON-filer

Man kan åpne JSON-filene til User og Workspace via kommandopaletten (`Ctrl+Shift+P`) ved:

```
Preferences: Open User Settings (JSON)
Preferences: Open Workspace Settings (JSON)
```

Disse er normalt små og oversiktlige. Disse inneholder bare verdier bruker har satt. Heller ikke extensions legger inn settinger her.

Man kan også få oversikt over default-verdier nøklene har via kommandopaletten ved

```
Preferences: Open Default Settings (JSON)
```

Denne filen er imidlertid kjempestor, så her må man søke etter bestemte nøkler (f.eks. nevnt i de to ovennevnte filene).

For å eksemplifisere: Anta User settings inneholder

```
"editor.fontFamily": "Fira Code, Consolas, 'Courier New' monospace"
```

Et søk etter **editor.fontFamily** i default settings viser at nøkkelen har denne default-verdien:

```
"editor.fontFamily": "'Droid Sans Mono', monospace"
```

Slik kan man enkelt eksperimenter, resette til default verdi osv. uten å rote noe til eller miste oversikt, i det minste med nøkler man kjenner litt til.

Husk også at man kan kommentere i JSON-filene ved:

```
// Midlertidig test
```

hvilket også kan bidra til oversikt og forståelse over tid.

Settingsvinduet

For å åpne settingsvinduet kan man klikke `Ctrl + ,`. Det første man ser er en full, dynamisk dokumentasjon av JSON-settingene, brukervennlig og oversiktlig. Her bør man lese og gjøre seg kjent. I tillegg kan man lete etter bestemte ord eller konkrete nøkler i søkefeltet, som vi straks skal se på.

Øverst ser man også en knapp **Backup and Sync Settings**. Her kan man sett opp en synkronisering av VS Code-innstillingene, slik at man kan ha samme arbeidsmiljø på flere PC-er. Da deles

- User settings (**settings.json**)
- Keybindings
- Snippets

- Themes, ikoner og andre preferences
- Extensions (installerte utvidelser og deres innstillinger)

Dette deles ikke:

- Workspace settings (**.vscode/settings.json**) synkes ikke.
- Folder / multi-root settings

Workspace-settings er spesifikke for hvert prosjekt, og normalt deles disse (**.vscode/settings.json**) via av et vanlig Git/GitHub-håndtert prosjekt.

En særlig nyttig søk i settingsvinduet er ved følgende filter:

```
@modified
```

Denne viser, både for User og Workspace, hvilke endringer bruker har gjort. I dette vinduet kan man så lese om nøkler, eksperimentere med innstillinger og i det hele tatt holde oversikt over det som gjerne angår en mest.

La oss som eksempel søker etter **editor.fontFamily** i søkefeltet. Da får man fram dokumentasjonen for denne nøkkelen. Man ser verdien nøkkelen har enten i User eller Workspace (velges rett under søkefeltet). I mitt tilfelle vises

```
Fira Code, Consolas, 'Courier New', monospace
```

for User og

```
'Droid Sans Mono', monospace
```

for workspace.

Her må vi stoppe opp og avklare noe som ofte forvirrer. Alene forteller ikke verdifeltet hvilken verdi nøkkelen faktisk har. Default-verdi vises nemlig om tilhørende nøkkelen ikke er satt. I eksempelet settes det følgende eksplisitt i User-json

```
"editor.fontFamily": "Fira Code, Consolas, 'Courier New', monospace"
```

mens Workspace-json ikke setter nøkkelen. Det betyr at font-settingen bruker har satt globalt faktisk gjelder prosjektet man jobber med. Men ut fra settingsvinduet (dokumentasjonsvinduet), siden Workspace overstyrer User, kan det se ut som at

```
"editor.fontFamily": "'Droid Sans Mono', monospace"
```

er aktiv, hvilket den ikke er.

Likevel, settingsvinduet er nyttig. Man kan sette verdier, og tilhørende JSON-fil oppdateres automatisk. Motsatt oppdatering skjer også. Videre har vi sett viktigheten av søkefilter `@modified`, og vi har i tillegg flere slike:

```
@modified:    – hva som er endret
@id:          – hopp direkte til en nøkkel
@ext:         – extensions
@lang:        – språkrelatert filtering  (MD, ADOC, ...)
@tag:         – filtrer på metadata-tags
@feature:     – VSCode-område (editor, terminal, git, filutforsker, ... )
@policy:      – settings som kan styres av organisasjon/policy
```

VS Code vil autofullføre disse (foreslå mulige fortsettelser der det er mulig), og man kan også kombinere dem:

```
@modified @id:editor.fontFamily
```

```
@modified @ext:markdown
```

```
@feature:editor @modified
```

Når man finner fram til aktuell nøkkel, kan man også klikk på tannhjulet ved siden av og resette verdi til default eller kopiere settingen ut (f.eks. i JSON-format).

Struktur i JSON Default settings

Selve filen inneholder ca. 11 000 linjer og kan synes vanskelig å få oversikt i. Men alt er hierarkisk organisert i “namespaces” gruppert med punktnotasjon. Her ser vi vi de øverste nivåene:

```
settings
 │
 ├──── editor
 │     ├──── font*
 │     ├──── cursor*
 │     ├──── format*
 │     ├──── suggest*
 │     ├──── minimap*
 │     ├──── wordWrap
 │     ├──── tabSize
 │     └──── render*
 │
 ├──── files
 │     ├──── autoSave
 │     ├──── encoding
 │     ├──── exclude
```

```
|     ├──── associations
|     └──── trim*
|
├──── workbench
|     ├──── colorTheme
|     ├──── iconTheme
|     ├──── startupEditor
|     ├──── activityBar*
|     ├──── editor*
|     └──── sideBar*
|
├──── window
|     ├──── zoomLevel
|     ├──── title*
|     └──── openFilesInNewWindow
|
├──── terminal
|     └──── integrated
|         ├──── font*
|         ├──── shell*
|         ├──── profiles*
|         ├──── scrollback
|         └──── cursor*
|
├──── explorer
|     ├──── confirmDelete
|     ├──── sortOrder
|     └──── compactFolders
|
├──── search
|     ├──── exclude
|     ├──── useIgnoreFiles
|     └──── followSymlinks
|
├──── git
|     ├──── enabled
|     ├──── autofetch
|     ├──── confirmSync
|     └──── decorate*
|
├──── debug
|     ├──── openDebug
|     ├──── inlineValues
|     └──── toolBarLocation
|
├──── extensions
|     └──── (extension-specific settings)
|
└──── [language-id]
    ├──── editor.*
    ├──── files.*
```

```
    └──── format*
```

der en JSON-notasjon som f.eks.

```
"editor.fontSize": 14
```

representeres ved

```
editor
  └──── fontSize
```

Systemer er dypere enn vist på flere steder. Eksempler som

```
"terminal.integrated.fontSize"
```

som ville vært representert av

```
settings
  └──── language override (json)
        └──── editor.tabSize
```

vises ikke i oversikten.

Viktige nøkler

Det er umulig, og ikke nødvendig, å ha oversikt over større deler av nøkkelsettet til VS Code. Her kommer imidlertid et utvalg som kan være aktuelt for relativt nye brukere.

1. Editor / redigering

```
NØKKEL                  HVA DEN GJØR                TYPISK VERDI
editor.fontFamily       Skrifttype i editor         "Fira Code", monospace
editor.fontSize         Størrelse på teksten        14, 16 …
editor.lineHeight       Linjeavstand                20, 1.5 …
editor.wordWrap         Om linjer skal brytes       "on", "off", "wordWrapColumn"
editor.tabSize          Hvor mange mellomrom per tab  2 eller 4
editor.insertSpaces     Tab er spaces eller ekte tab  true / false
editor.formatOnSave     Autoformattering ved lagring  true / false
editor.minimap.enabled  Viser eller skjuler minimap   true / false
```

2. Språkspesifikke innstillinger

```
NØKKEL                    FUNKSJON          TYPISK VERDI
editor.quickSuggestions   Skrive forslag    true/false
editor.formatOnSave       Autoformatering   true
editor.tabSize            Tab-størrelse     2/4
```

3. Filer

```
NØKKEL                        FUNKSJON               TYPISK VERDI
files.autoSave                Autolagring            "afterDelay", "onFocusChange"
files.exclude                 Skjuler filer          {"**/.git": true}
files.trimTrailingWhitespace  rm whitespace on save  true
files.insertFinalNewline      Newline på slutten     true
files.encoding                Filkoding              "utf8"
```

4. Editor / display / UI

```
NØKKEL                  FUNKSJON
workbench.colorTheme    Tema (dark/light)
workbench.iconTheme     Ikonpakke for filer
editor.renderWhitespace Viser spaces/tabs
editor.cursorBlinking   Blinking cursor
editor.lineNumbers      Linjenummer visning
```

5. Linting / formatering / Markdown / AsciiDoc

```
NØKKEL                        FUNKSJON
markdownlint.config           Konfig for markdownlint
markdown.preview.breaks       Hard line breaks i preview
markdown.preview.fontFamily   Font i preview
asciidoc.preview.fontFamily   Font i preview
asciidoc.preview.breaks       Hard line breaks
asciidoc.extension.path       Tilpass path til Asciidoctor binær
```

6. Git

```
NØKKEL                  FUNKSJON
git.enableSmartCommit   Commit uten å skrive melding
git.autofetch           Henter automatisk
git.confirmSync         Bekreft før sync
```

7. Terminal / konsoll

```
NØKKEL                              FUNKSJON
terminal.integrated.fontFamily      Terminal font
terminal.integrated.fontSize        Terminal font size
terminal.integrated.scrollback      Hvor mange linjer som huskes
terminal.integrated.cursorBlinking  Cursor blinking
```

Extensions

Også når det gjelder extensions kan eksperimentering, mer eller mindre gode tips etc, over tid føre til redusert oversikt. Så det første man trenger er å vite hvordan man kan få litt oversikt over installerte extensions, samt hvilke som er aktive eller ikke.

Man kan åpne extensions-panelet grafisk eller ved

```
Ctrl + Shift + X
```

Her vil en serie extensions listes, både de man allerede har og mange andre som tilbys. Man kan filtrere etter følgende valg ved å klikke på filtreringsikonet til høyre i søkefeltet.

```
- Installed   → alt du har
- Updates     → hvilke som kan oppdateres
- Built in    → innebygde extensions
- Enabled     → faktisk aktive
- Disabled    → installert men ikke i bruk
```

Man kan også søke på bestemt ting, f.eks. knyttet til C++, Python, Markdown, Asciidoc etc som f.eks:

```
@installed markdown
```

```
@installed asciidoc
```

I tillegg kan man fra **Command Palett** (CTR+SHIFT+P) taste inn

```
Extensions: Show Running Extensions
```

Uansett kan man da se hvilke utvidelser som er aktive (enabled) eller ikke-aktive (disabled), og man kan velge/endre dette i vinduene som dukker opp når man klikker på utvidelsene.

Det er ikke uvanlig å ha endt opp i en situasjon der konkurrerende utvidelser kjører (der flere tilbyr samme tjenester), så det kan være greit å rydde litt opp her innimellom, prøve hvilke man er mest fornøyd med, og deaktivisere resten (evt avinstallere dem om man virkelig har bestemt seg).

Task automation

For de som er interessert i spesielle ting som Python, C++ eller hva, fins det automation-tips der ute. Her skal vi vise et mer generelt eksempel på hvordan automatiserer kjøring av et shell-skript kalt **runner.sh** på prosjektkatalogen.

Det er relativt enkelt å sette opp *task automation*. I VS Code gjør man:

1. Åpne prosjektmappen
2. Trykk `Ctrl+Shift+P`
3. Skriv: **Tasks: Configure Task**
4. Velg: **Create tasks.json file from template**
5. Velg: **Others**

Dette genererer filen

```
.vscode/tasks.json
```

som nå åpnes i VS Code. Rediger innholdet til:

```
{
  "version": "2.0.0",
  "tasks": [
    {
      "label": "Passende beskrivelse",
      "type": "shell",
      "command": "./runner.sh",
      "group": "build",
      "problemMatcher": []
    }
  ]
}
```

Tasken kan nå kjøres ved `Shift+Ctrl+B` og klikke på tasken med beskrivelsen du valgte.

Linjene

```
"group": "build",
"problemMatcher": []
```

er nyttige å legge til. Den første gjør at VS Code hopper rett til `Run Task` etter `Shift+Ctrl+B`, mens den andre gjør at eventuell output ikke tolkes som kompilator-feil.

Skal man siden automatisere flere tasks, legges disse inn i samme fil.

Tips: Det fins en extension Task som gjør at man kjøre tasken via en tekst-knapp nede i taskbar i VS Code, for enda kortere snarvei. Søk gjerne etter informasjon om dette.

Command Pallette

Vi har benyttet kommandopaletten (som er svært nyttig) flere ganger. Her skal vi supplere med en totaloversikt over hva man kan gjøre.

For det første er oppgavene på formen:

```
[System / Modul]: [Handling]
```

som f.eks.

```
Preferences: Open User Settings (JSON)
```

Man trenger ikke å skrive hele kommandoen. I praksis en gjør VS Code en fuzzy search blant alle kommandoer.

En tre-representasjon av palettens muligheter ser slik ut:

```
Command Palette
 │
 ├──── File
 │     ├──── New File
 │     ├──── Open File
 │     ├──── Save
 │     ├──── Save All
 │     └──── Reopen Closed Editor
 │
 ├──── Edit
 │     ├──── Undo / Redo
 │     ├──── Format Document
 │     ├──── Rename Symbol
 │     ├──── Refactor
 │     └──── Emmet Expand
 │
 ├──── View
 │     ├──── Toggle Sidebar
 │     ├──── Toggle Panel
 │     ├──── Appearance
 │     │     ├──── Change Theme
 │     │     ├──── Change Icon Theme
 │     │     └──── Toggle Full Screen
 │     └──── Open View (Explorer, Search, SCM, Run)
 │
 ├──── Go / Navigation
```

```
│     ├──── Go to File
│     ├──── Go to Symbol
│     ├──── Go to Definition
│     ├──── Go to Line
│     └──── Go Back / Forward
│
├──── Search
│     ├──── Find in File
│     ├──── Replace
│     ├──── Search in Workspace
│     └──── Toggle Regex / Case / Whole Word
│
├──── Terminal
│     ├──── Create New Terminal
│     ├──── Split Terminal
│     ├──── Select Default Profile
│     └──── Run Active File
│
├──── Tasks
│     ├──── Run Task
│     ├──── Configure Task
│     ├──── Restart Running Task
│     └──── Terminate Task
│
├──── Debug
│     ├──── Start Debugging
│     ├──── Add Configuration
│     ├──── Select Debug Configuration
│     └──── Toggle Breakpoint
│
├──── Git / Source Control
│     ├──── Commit
│     ├──── Push
│     ├──── Pull
│     ├──── Checkout Branch
│     └──── Stage Changes
│
├──── Extensions
│     ├──── Install Extensions
│     ├──── Show Installed Extensions
│     ├──── Disable Extension
│     └──── Reload Window
│
├──── Preferences
│     ├──── Open Settings (UI)
│     ├──── Open Settings (JSON)
│     ├──── Keyboard Shortcuts
│     ├──── Profiles
│     └──── Settings Sync
│
└──── Developer
```

```
    ├──── Reload Window
    ├──── Toggle Developer Tools
    ├──── Inspect Context Keys
    └──── Show Running Extensions
```

Det kan være lurt å gjøre seg kjent her for effektivt arbeid.

Andre ting å fordype seg i

VS Code har flere muligheter for effektivisering av arbeid. Hva man vil satse på avhenger av type prosjekt, smak og annet. Her vil vi bare nevne ting man bør søke opp dersom begynner å jobbe mer arbeidsintensivt.

- Snippets

  Snippets er snarveier til å sette inn tilpassede kodeblokker eller annen fast tekststruktur, gjerne med placeholders for ulike valg eller variable. Disse settes opp globalt (i user, ikke i workspace).

  Merk: Noen grunnleggende snippets følger med VS Code. Og om man markerer et ord, kan man velge i et utvalg om man klikker på lyspæren, den såkalte **lightbulb**.

  Merk: Lightbulb kommer av og til med effektive forslag ellers også, så eksperimenter gjerne.

- Multi cursor editing

  Multi cursor editing gjør det mulig å kjapt editere flere steder i teksten samtidig. Mulighetene er mange, så jeg anbefaler å lese mer om dette.

- Snarveier

  Man kan enkelt sette opp snarveier (keybindings) for ofte utførte oppgaver. Søk gjerne opp informasjon om dette.

- Profiles

  Profiler gir et separat arbeidsmiljøer inni VS Code. Dvs. at man kan samle **Settings** Extensions **Keybindings** Snippets **Layout** Tasks

i en profil og bytte mellom ulike profiler etter behov. Dermed kan man ha bare de utvidelsene man trenger, de snarveiene man trenger osv. for enda mer strømlinjeformet arbeidsflyt, uten kollisjoner og unødvendige ting. Søk opp mer informasjon om dette.