

# Avhengigheter (i prosjektkoden)

# Agenda

- ▶ Avhengigheter
  - ▶ Eksplisitte og Naive
- ▶ Problemer med avhengigheter
- ▶ Byggeverktøy
- ▶ Versjonsnumre
- ▶ Automatisk oppgradering av versjoner

# Avhengigheter

- ▶ Det er sjeldent at programvare utvikles fullstendig fra bunnen av
  - ▶ Vi bygger på/gjenbruker typisk noe ferdigutviklet kode
  - ▶ Rammeverker, biblioteker, moduler, osv.
- ▶ Bruk av slik ferdigutviklet kode kalles avhengigheter
- ▶ Fordel - Slipper å finne opp hjulet på nytt (mindre ressurser)
- ▶ Ulemper
  - ▶ Vi har sjeldent kontroll/oversikt over alt i avhengighetene
  - ▶ Hvis noe går galt med avhengighetene ødelegger det ofte koden vår



**Randall Koutnik**  
@rkoutnik

Why programmers like cooking: You peel the carrot, you chop the carrot, you put the carrot in the stew. You don't suddenly find out that your peeler is several versions behind and they dropped support for carrots in 4.3

12:22 PM - 16 Jan 2019

# Bibliotek vs. Rammeverk

## Bibliotek

- ▶ Samling med klasser
  - ▶ Kan tenkes på som en samling med verktøy
- ▶ Vår kode bruker/kaller biblioteket for å oppnå funksjonalitet enklere/raskere
- ▶ Trenger typisk ikke direkte tilgang til I/O
- ▶ Typiske eksempler:
  - ▶ Serialisering av objekter
  - ▶ Databasekommunikasjon
  - ▶ Logging
  - ▶ Samling med algoritmer

## Rammeverk

- ▶ Samling med klasser som vi skriver programmet **rundt**
  - ▶ Det typiske eksemplet er et rammeverk for brukergrensesnitt
- ▶ Rammeverket bruker/kaller *vår* kode
  - ▶ Vi tilpasser rammeverket til vårt problem
- ▶ Starter og styrer ofte «hele» programmet - I/O
- ▶ Vi blir fort veldig avhengige av rammeverket
  - ▶ Typisk arv av rammeverkets klasser

# Eksplisitte avhengigheter

- ▶ Eksplisitte avhengigheter - Avhengigheter vi definerer og kan verifisere i prosjektet vårt
- ▶ For eksempel
  - ▶ Bibliotek for database-kommunikasjon
  - ▶ Bibliotek for håndtering av bilder og videoer
  - ▶ HTTP-klient
  - ▶ Modul med sorteringsalgorimer
  - ▶ Rammeverk for mobil-applikasjoner
  - ▶ Testrammeverket

# Naive avhengigheter

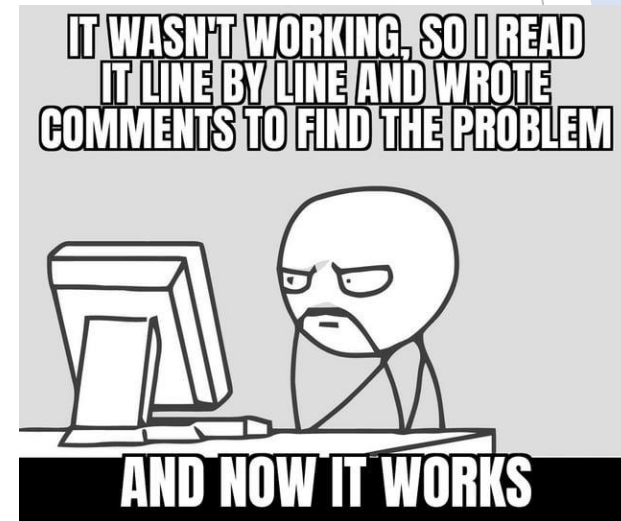
- ▶ Naive avhengigheter - Avhengigheter vi kanskje ikke tenker like mye på
- ▶ For eksempel
  - ▶ Windows / Linux
  - ▶ Java, .NET, C#, Python, Javascript, PHP
  - ▶ MySQL
  - ▶ Google Cloud / Microsoft Azure / Amazon Web Services
  - ▶ osv...

# Hvorfor bruke avhengigheter

- ▶ Oftest benytter vi avhengigheter fordi vi selv velger det
  - ▶ Fordelaktig på et vis
  - ▶ Kanskje helt nødvendig for å utvikle produktet
- ▶ Men avhengigheter kan også påvirkes av ikke-tekniske ting
  - ▶ Krav
    - ▶ Produktet skal være laget i ...
    - ▶ Det skal være kompatibelt med ...
  - ▶ Rettigheter / kostnad / tid
    - ▶ «Vi har bare lisens til ...»
    - ▶ «Det har vi ikke råd til»
    - ▶ «Vi har ikke tid til å utvikle dette selv ...»

# «Men det funker jo på ...»

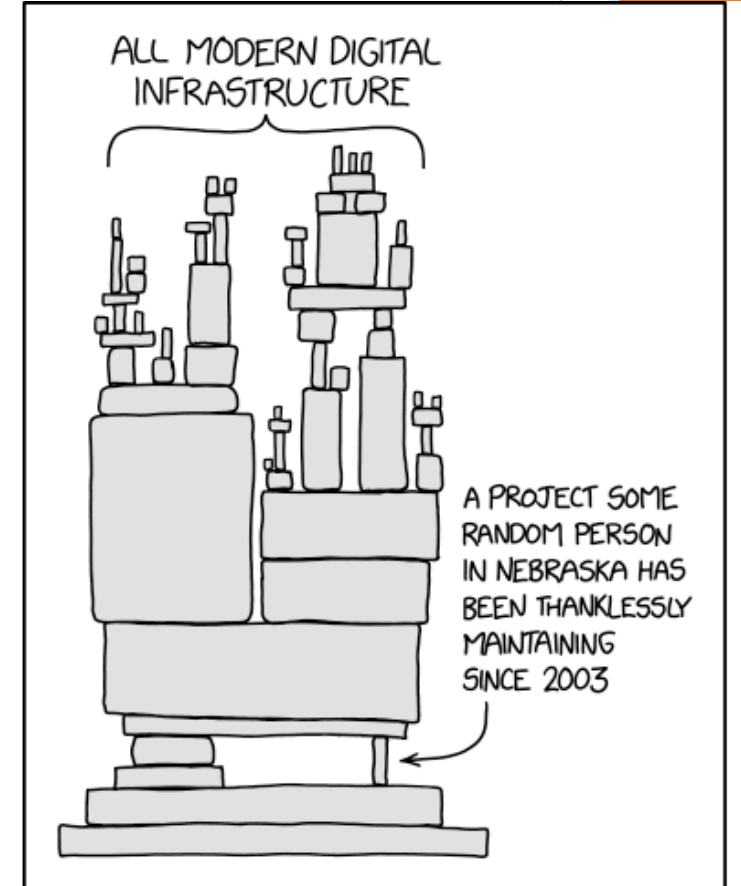
- ▶ Hvor ofte har du hørt (eller sagt) ...?
  - ▶ «Det funker på min maskin»
  - ▶ «Det funka når jeg testa sist»
  - ▶ I SE er det ofte - «Det funker på test-serveren»
- ▶ Dette kommer nesten alltid av noe annet enn koden selv
  - ▶ Avhengigheter kan være én årsåk
    - ▶ Forskjellig versjon/type/generasjon e.l.
- ▶ Du er egentlig heldig hvis du blir klar over problemet...





# Dependency hell

- ▶ Jo flere avhengigheter du har jo større sannsynlighet for kontinuerlige problemer
  - ▶ Spesielt når avhengighetene selv har avhengigheter (til hverandre)
  - ▶ Dette kalles dependency hell
  - ▶ Uendelig slit for å oppnå en (ofte) midlertidig løsning
- ▶ Kan oppstå på mange måter
  - ▶ Avsluttet støtte for biblioteker
  - ▶ Dårlig skrevet kode i avhengigheter
  - ▶ Dårlig dokumentasjon → Feil bruk
  - ▶ osv...
- ▶ Altså - Vi bør ta grep for å unngå å havne her
  - ▶ Unngå undøvendige avhengigheter
  - ▶ Oversikt og standardisering



<https://xkcd.com/2347/>

# Byggeverktøy / avhengighetssystemer

- ▶ Vi kan benytte byggeverktøy for å eksplisitt definere avhengigheter med versjon på en standardisert måte
  - ▶ Slipper manuell håndtering av avhengighetene
  - ▶ Gjelder for alle utviklere i prosjektet
- ▶ **Java:** Maven, Gradle
- ▶ **Python:** pip, poetry, pipenv
- ▶ **PHP:** composer
- ▶ **C#:** NuGet
- ▶ **Node.js:** npm, yarn
- ▶ Vi har altså tidligere lagt til JUnit og Mockito som dependencies via Maven sin pom.xml

# Maven - pom.xml

- ▶ Vi legger til dependencies som xml-tagger
  - ▶ Under dependencies-taggen
  - ▶ dependency - representerer én avhengighet
  - ▶ groupId - «navn» på dependency-utviklerne
  - ▶ artifactId - navnet på dependency-en
  - ▶ version - dependency-ens spesifikke versjon
  - ▶ scope - definerer hvilke deler av prosjektets livssyklus dependency-en er relevant for
- ▶ Våre egne prosjekter definerer de samme taggene og kan dermed benyttes som dependencies i andre prosjekter
  - ▶ ... men blir ikke automatisk tilgjengelig for nedlastning over nett
  - ▶ Vi kaller ofte slike prosjekter (våre egne og andre avhengigheter vi integrerer) for «moduler»

```
<dependencies>  
  <dependency>  
    <groupId>org.junit.jupiter</groupId>  
    <artifactId>junit-jupiter</artifactId>  
    <version>5.10.2</version>  
    <scope>test</scope>  
  </dependency>  
</dependencies>
```

```
<?xml version="1.0" encoding="UTF-8"?>  
<project xmlns="http://maven.apache.org/POM/4.0.0"  
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0  
    http://maven.apache.org/maven-v4_0_0.xsd">  
  <modelVersion>4.0.0</modelVersion>  
  
  <groupId>org.example</groupId>  
  <artifactId>UnitTestingIntro</artifactId>  
  <version>1.0-SNAPSHOT</version>
```

# Bruk av interne avhengigheter

- ▶ Vi kan benytte interne prosjekter som avhengigheter
  - ▶ Fungerer akkurat som med eksterne avhengigheter
- ▶ Vi kan altså bruke interne biblioteker på tvers av prosjekter og teams hvis vi ønsker
- ▶ Dette er typisk mindre «skummelt» enn eksterne avhengigheter ettersom vi mye har større kontroll

# Versjonsnumre

Monday, August 30, 2021

Python 3.9.7 and 3.8.12 are now available

Python 3.9.7

Get it here: <https://www.python.org>

Python 3.9.7 is the newest major version, and Python 3.9.6 which is a similar amount of time ago.

PHP 8.1.0 RC 3 available for testing »

The PHP team is please  
the rough outline of wh

For source downloads c



Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine.

Download for Windows (x64)

14.18.0 LTS

Recommended For Most Users

### 16.10.0 Current

## Latest Features

[Other Downloads](#) | [Changelog](#) | [API Docs](#)[Other Downloads](#) | [Changelog](#) | [API Docs](#)

Or have a look at the Long Term Support (LTS) schedule

# Versjonsnumre

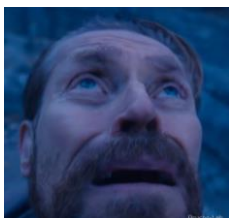
- ▶ Versjonsnumre sier oss i utgangspunktet ganske lite (Uintuitive)
- ▶ Det er opp til hvert enkelt prosjekt å definere dem (ikke alltid standardisert)
  - ▶ Chrome sin siste versjon er 127  
Siste versjon for windows 8 var 109...

"browser": "0.3.2",

"brfs": "^1.4.3",

"clean-css": "~2.0.7",

"clone": "0.1.16",



Release ↕	General availability ↕	Latest minor version ↕
5.1 LTS	14 November 2008; 15 years ago <sup>[48]</sup>	5.1.73 <sup>[49]</sup>
5.5 LTS	3 December 2010; 13 years ago <sup>[50]</sup>	5.5.62 <sup>[51]</sup>
5.6 LTS	5 February 2013; 11 years ago <sup>[52]</sup>	5.6.51 <sup>[53]</sup>
5.7 LTS	21 October 2015; 8 years ago <sup>[54]</sup>	5.7.44 <sup>[55]</sup>
8.0 LTS	19 April 2018; 6 years ago <sup>[56]</sup>	8.0.39 <sup>[57]</sup>
8.1 IR	18 July 2023; 12 months ago <sup>[58]</sup>	8.1.0 <sup>[59]</sup>
8.2 IR	25 October 2023; 9 months ago <sup>[60]</sup>	8.2.0 <sup>[61]</sup>
8.3 IR	16 January 2024; 6 months ago <sup>[62]</sup>	8.3.0 <sup>[63]</sup>
8.4 LTS	30 April 2024; 3 months ago <sup>[64]</sup>	8.4.2 <sup>[65]</sup>
9.0 IR	1 July 2024; 41 days ago <sup>[66]</sup>	9.0.1 <sup>[67]</sup>

<https://en.wikipedia.org/wiki/MySQL>

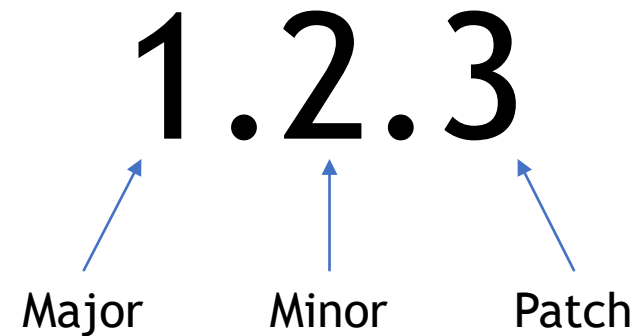
Hva skjedde med MySQL 6 og 7???

# Versjonsnumre - Semantisk Versjonering 2.0.0

- ▶ Semantisk versjonering (SemVer) gir et sett med regler for å kommunisere versjonering av **biblioteker**
  - ▶ Standardiserer format og mening
  - ▶ Definerer hva endringer betyr for de som er avhengig
  - ▶ Blir brukt av mange
- ▶ Deles opp i
  - ▶ Patch - Ingen nye features. Bare bugfixes
  - ▶ Minor - Ny men backwards-compatible funksjonalitet
    - ▶ Backwards-compatible: Vi trenger ikke fixe noe etter oppdatering
  - ▶ Major - Hvis endringer ikke er backwards-compatible
    - ▶ F.eks. Fjerner offentlige metoder eller endrer (bruk av) metoder
    - ▶ Teoretisk sett det eneste som er skummelt å oppdatere

1.2.3

Major Minor Patch



# Versjonsnumre - Semantisk Versjonering 2.0.0

- ▶ Major version zero (0.y.z)
  - ▶ Burde være reservert til utvikling
  - ▶ Kan endres når som helst
  - ▶ Public API-er med slike versjonsnumre burde IKKE anses som stabile
- ▶ Etter en versjon er blitt sluppet ut
  - ▶ En sluppet ut versjon skal ALDRI modifiseres
  - ▶ Hver endring SKAL få en ny versjon



# Automatisk oppgradering av biblioteker

- Det finnes syntax i Maven for automatisk oppgradering av biblioteker:

Rekkevidde	Betyr
1.0	En «soft requirement» på versjon 1.0. Maven kan overstyre om nødvendig.
[1.0]	En «hard requirement» på versjon 1.0
(,1.0]	Så lenge versjon $\leq 1.0$
[1.2,1.3]	Så lenge $1.2 \leq \text{versjon} \leq 1.3$
[1.0,2.0)	Så lenge $1.0 \leq \text{versjon} < 2.0$ . Vil altså holde oppdatert så lenge det bare er oppdateringer innen 1.0. Ingen major oppdatering.
[1.5,)	Så lenge versjon $\geq 1.5$ . Altså fra 1.5 og videre.
(,1.0],[1.2,)	Så lenge versjon $\leq 1.0$ eller versjon $\geq 1.2$ . Altså er alt mellom 1.0 og 1.2 ikke tillat.
(,1.1),(1.1,)	Så lenge versjon $< 1.1$ eller versjon $> 1.1$ . Altså er 1.1 ikke tillat.

<https://www.baeldung.com/maven-dependency-latest-version>

- Syntaks for lignende er unikt per byggeverktøy

# Automatisk oppgradering av biblioteker

- ▶ I Java (Maven) er det generelt anbefalt å holde bibliotek-versjoner statiske
- ▶ Ulemper med automatisk oppgradering:
  - ▶ Selv små bugfixes og nye features kan ødelegge
    - ▶ Det hjelper å ha automatiske tester, men likevel...
  - ▶ Sikkerhetsproblemer
    - ▶ Nye «features»
    - ▶ Ny funksjonalitet = nytt sikkerhetshull?
- ▶ En god tommelfingerregel er å holde versjoner statiske inntil man trenger en bugfix eller ny feature

## Software

### How one developer just broke Node, Babel and thousands of projects in 11 lines of JavaScript

Code pulled from NPM – which everyone was using

[https://www.theregister.com/2016/03/23/npm\\_left\\_pad\\_chaos/](https://www.theregister.com/2016/03/23/npm_left_pad_chaos/)

## Security

### Check your repos... Crypto-coin-stealing code sneaks into fairly popular NPM lib (2m downloads per week)

Node.js package tried to plunder Bitcoin wallets

[https://www.theregister.com/2018/11/26/npm\\_repo\\_bitcoin\\_stealer/](https://www.theregister.com/2018/11/26/npm_repo_bitcoin_stealer/)

# Generelle tips til avhengigheter

- ▶ Begrens avhengigheter der det er mulig
- ▶ Kan vi gjøre det selv?
  - Hvor lang tid vil det ta?
  - Hvor vanskelig er det?
- ▶ Typisk fornuftige avhengigheter
  - Databasekommunikasjon
  - Konvertering mellom datatyper (f.eks. objekt til/fra JSON)
  - Testing / Mocking
  - GUI-rammeverk