Utvecklingsmetodik

En rejäl crash course i versionskontroll, pakethantering, testning och paketering

Johan Holmberg 2019-05-07

MALMÖ UNIVERSITET

Dagens föreläsning

- Versionshantering
- Pakethantering
- Paketering
- Testning

MALMÖ UNIVERSITET

Metodik

But why?

MALMÖ UNIVERSITET

Mjukvaruutveckling är dyrt!

- Våra verktyg är dyra
- Våra miljöer är dyra
- Vi är dyra
- **Våra fel** är dyra

Felen vi gör resluterar i:

MALMÖ UNIVERSITET

Produktionsbortfall

MALMÖ UNIVERSITET

Tappad försäljning

Biljettkaoset: Kunder uppmanades åka gratis

SJ Publicerad 6 okt 2017 kl 08.57

Många försökte förgäves köpa biljetter på Stockholms centralstation på fredagen. Men vid lunchtid …

Foto: LEIF BRÄNNSTRÖM

MALMÖ UNIVERSITET

Olyckor

MALMÖ UNIVERSITET

Dödsfall

MALMÖ UNIVERSITET

Eller värre...

MALMÖ UNIVERSITET

Ja, men varför?

MALMÖ UNIVERSITET

MALMÖ UNIVERSITET

Kodapan

Skriver kod. That’s it.

Ett kugghjul i ett maskineri:

- Skriver kod mekaniskt
- Saknar överblick
- Utan egentligt ansvar
- Väldigt utbytbar

Cowboy-kodaren

Skriver kod. That’s it.

En “fri själ”, men inte så bra:

- Ofta orutinerad
- Ostrukturerad
- Saknar kvalitetstänk
- Inte hållbar

Programmeraren

Skriver bra kod. Programmerar.

En renässansmänniska:

- Skapande
- Konstnärlig
- Ensamt geni
- Inte hållbar i längden

Utvecklaren

Skapar lösningar. Utvecklar.

En ingenjörstyp:

- Metodisk och strukturerad
- Grupporienterad
- Kvalitetstänkande
- Hållbar

Hur blir vi ordentliga utvecklare?

Genom att arbeta strukturerat och rationellt!

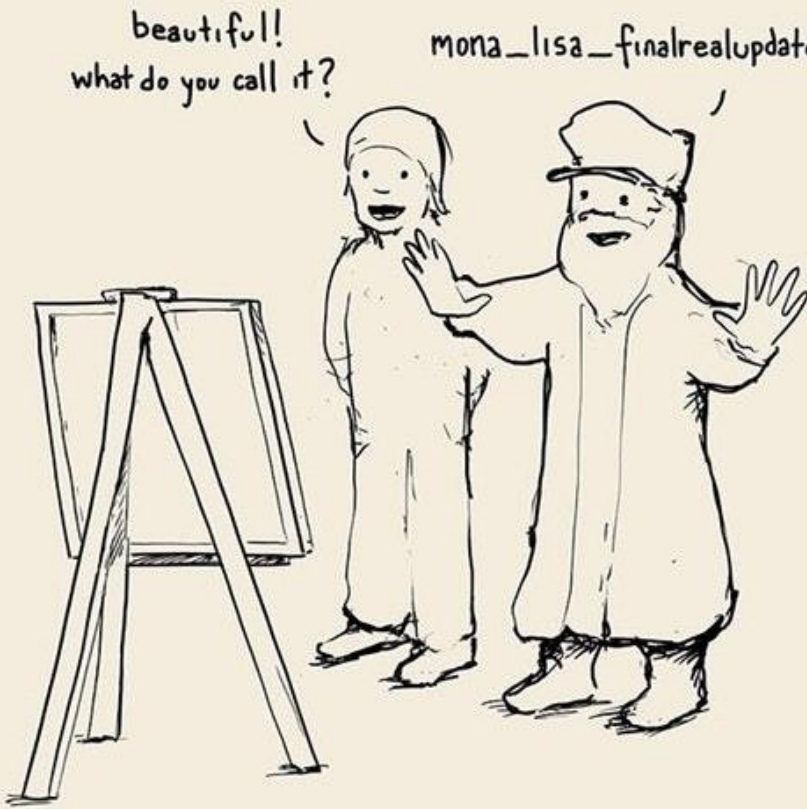
- Tänk först – skriv kod sedan. Gör vi rätt saker?
- Kan vi återanvända tidigare skriven kod? Hur?
- Kan vi effektivt samarbeta kring vår mjukvara?
- Hur dokumenterar vi?
- Hur säkerställer vi att koden fungerar?

MALMÖ UNIVERSITET

Versionshantering

Vårt problem

- Riktiga mjukvaruprojekt är stora
- Riktiga mjukvaruprojekt kommer att innebära flera – och parallella – releaser av samma mjukvara
- Hur hanterar vi detta på ett effektivt sätt?



Vad är versionshantering?

Versionshantering är den metod med vilken vi kan hantera flera parallella versioner av en mjukvara

- Möjliggör samtidig utveckling av olika delar av mjukvaran
- Ger oss möjligheter att känna till – och dra nytta av – tidigare versioner av mjukvaran
- Möjliggör samtidig utveckling av nischade versioner eller utgåvor av en mjukvara

MALMÖ UNIVERSITET

Fördelarna med versionshantering

- EN SANNING – Vi kan alla vara överens om vilken kod som är den aktuella
- Samtidig utveckling
  - Gör det enklare att dela och återanvända kod
  - Olika team kan arbeta på samma projekt
- Historia och spårbarhet
  - Återgå till tidigare versioner av ett projekt
    - Förstörde din kursare din kod? Ingen fara, den finns kvar!
  - Underhåll flera olika releaser av samma fil, modul eller applikation
  - Vem gjorde vad? Nu kan vi veta!
- Minskade risker
  - Ha inte all kod på en enda dator
  - Låt inte vem som helst förändra koden

MALMÖ UNIVERSITET

Viktiga begrepp

- **Versioner** eller **revisioner** – en uppsättning förändringar till källkoden.
- **Brancher** eller **grenar** – ett tillfälligt arbetsspår som lever parallellt med huvudarbetsspåret.
- **Trunk** eller **master** – projektets huvudarbetsspår. Här lever den “senaste”, “riktiga” versionen av mjukvaran.
- **Merge** eller **sammanslagning** – en punkt där två eller flera grenar slås samman till en.
- **Konflikter** – uppstår när versionshanteringsmjukvaran inte själva kan lösa en sammanslanging.
- **Resolve** – den manuella handpåläggning som utförs för att lösa en konflikt.
- **Commit** eller **incheckning** – den händelse som skapar en ny revision av koden.

Viktiga begrepp

- **Repository** – den plats där all kod och historik finns sparad. Kan vara på en extern server, men kan även vara lokal.
- **Intialisering** – skapandet av ett nytt repository.
- **Working copy** eller **arbetskopia** – den ögonblickskopia av koden som du just nu arbetar med. Fram tills att den checkas in i ett repsitory påverkar den ingen annan.
- **Checkout** – skapar en arbetskopia av en specifik revision av koden.
- **Push/pull** – kopiering av ändringar mellan två olika repositories. En **push** innebär att du skickar en ändring, en **pull** innebär att du hämtar en ändring.
- **Kloning** – den händelse som skapar en exakt kopia – en klon – av ett annat repository.

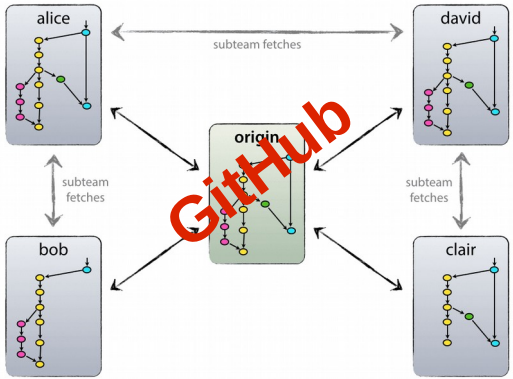
MALMÖ UNIVERSITET

- Det för tillfället flitigast användna verktyget för versionshantering
- Välprövat
  - Hanterar ohemult stora projekt (exempelvis Linux)
  - Snabbt!
- Hanterar arbetsflöden på ett bra sätt
  - Byggt för att vara bra på att dela upp projekt och slå samman dem igen
    - Fork/pull request och branch/merge
- Distribuerat – ingen central server krävs
  - Pull/Push
- Open Source!

Git: https://git-scm.com

MALMÖ UNIVERSITET

- En tjänst för lagring och delning av Git-repositories
- Stor i open source-världen
- Erbjuder även kringtjänster såsom wikis och viss ärendehantering



Github: https://github.com

MALMÖ UNIVERSITET

Demo

Johan leker med git

MALMÖ UNIVERSITET

Pakethantering

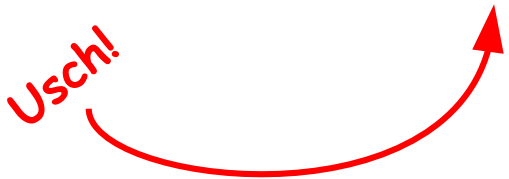
Vårt problem

- Riktiga mjukvaruprojekt är stora
- Riktiga mjukvaruprojekt innehåller oftast externt skriven mjukvara
- Hur hanterar vi detta på ett effektivt sätt?

Vad är pakethantering?

Modulbaserad mjukvara behöver kontrolleras – detta görs med fördel med en pakethanterare

- Återanvänd din (och andras) kod
- Separation of concerns
- Slipp ifrån jobbet med att dra ner paket manuellt

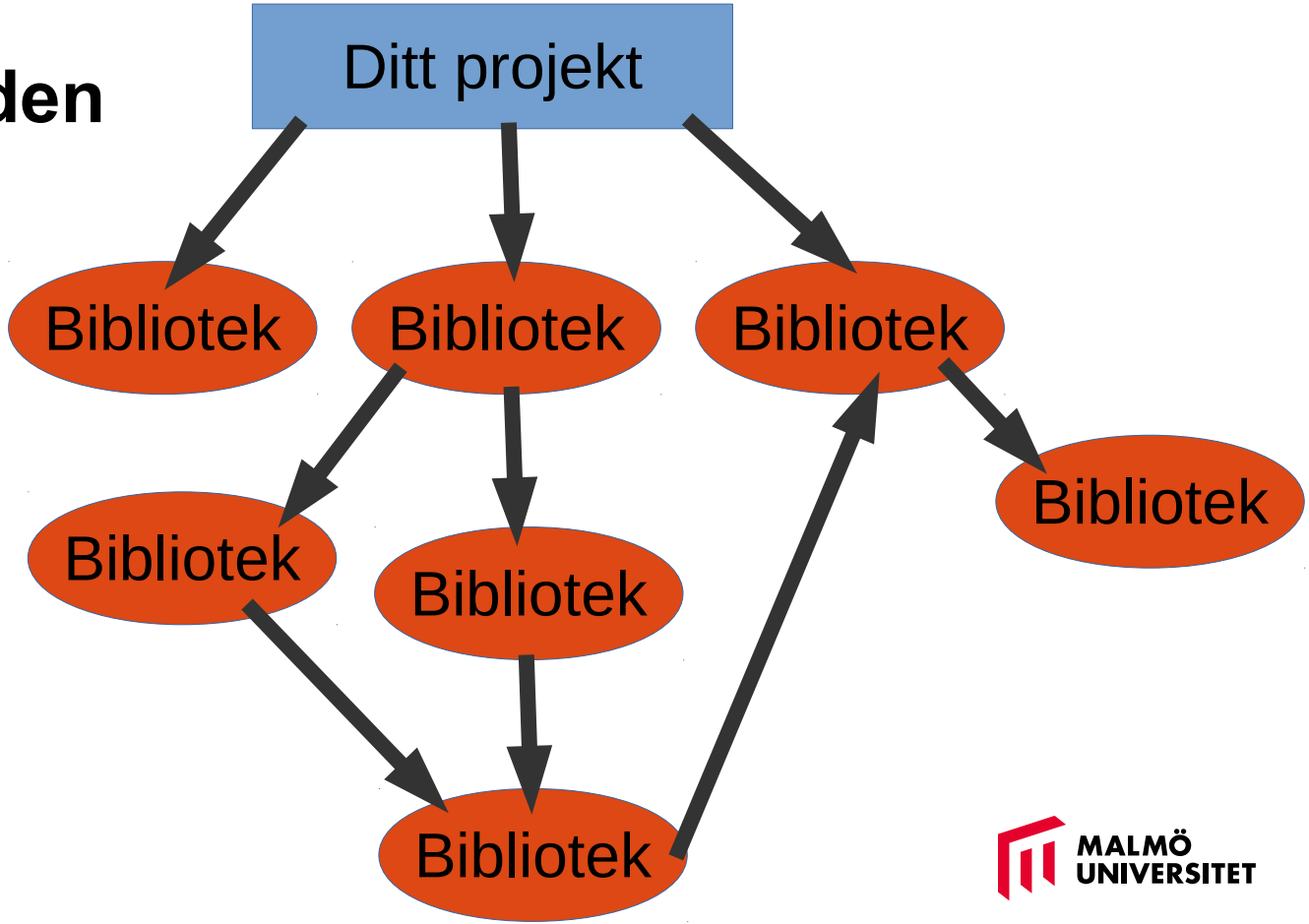


MALMÖ UNIVERSITET

Några viktiga termer

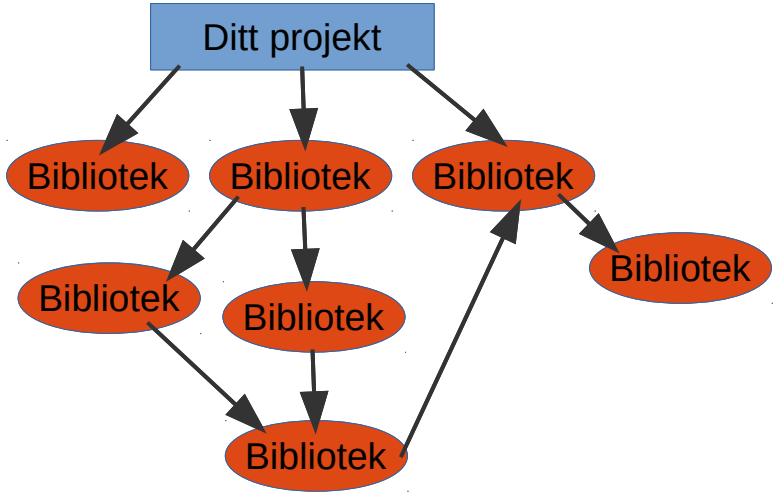
- **Paket**: En samling av filer som levereras tillsammans. Kan vara körbara filer, bibliotek eller andra resurser, så som media, typsnitt eller konfigurationsfiler. I JavaScript kallas dessa allmänt för **moduler**.
- **Pakethanterare**: En mjukvara som hanterar paket.
- **Repository** (eller **repo**): En plats där paket kan publiceras, sökas efter och laddas ner från.
- **Beroende**: Ett eller flera paket som behövs för att ett specifikt paket ska kunna användas.
- **Beroendekonflikt**: Ett tillstånd som uppstår då två paket beror på samma paket.
- **Version**: En specifik utgåva av ett specifikt paket. Genom att känna till versionsinformation kan en pakethanterare hålla koll på flera versioner av ett paket, vilket minskar risken för beroendekonflikter.

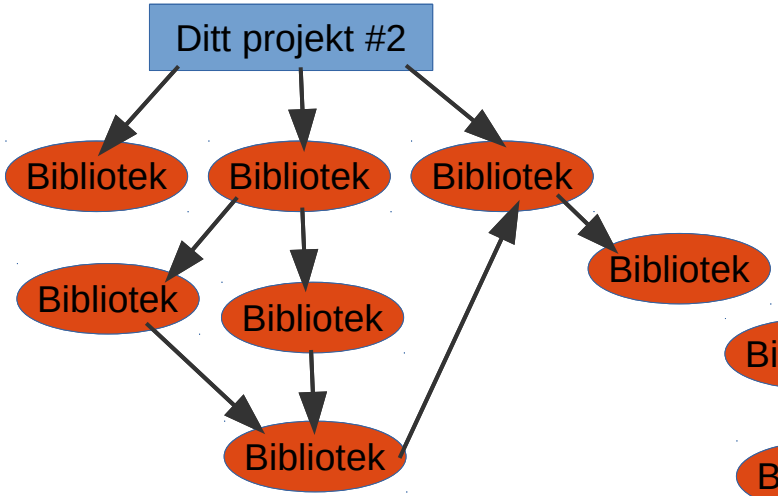
Beroenden

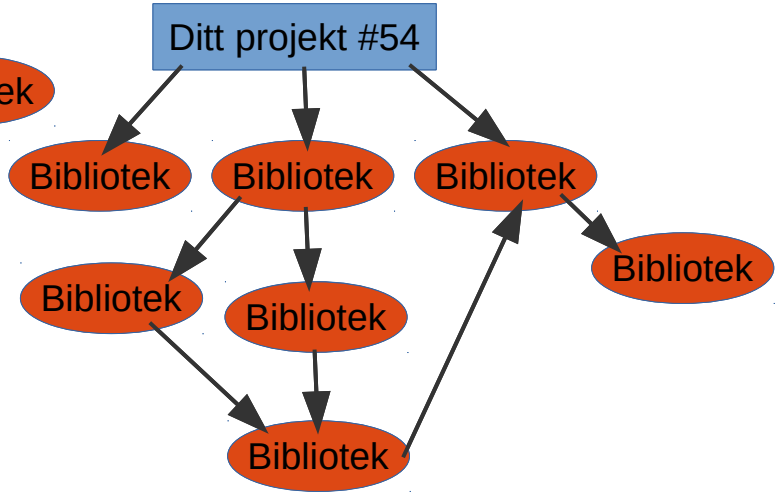


MALMÖ UNIVERSITET

Beroenden







MALMÖ UNIVERSITET

Beroendehantering i mjukvaruprojekt

- Python: pip
- Java: Maven/Gradle
- PHP: Composer
- C#/.NET: NuGet
- **Javascript: npm**

Varför beroendehantera i mjukvaruprojekt?

- Robustare projekt och mjukvara
- Enklare att starta nya projekt
- Enklare för nya utvecklare att börja arbeta
- Mindre kod i era kod-repositories

MALMÖ UNIVERSITET

Pakethanteraren npm

- De facto-standard för vettig beroendehantering i JavaScript
- Bygger på Node.js – npm (Node Package Manager) är dess pakethanterare
  - Bygger på en kodstandard som heter CommonJS
  - Går att använda till annat än just Node.js
- Sköter installation av moduler åt dig
  - Du behöver inte längre versionshantera och leverera alla beroenden – bara en fil som berättar vilka paket som krävs

Pakethanteraren npm

- Hanterar beroenden i flera led
- Kan skilja mellan olika versioner av moduler
- Förenklar uppdatering av mjukvaran när underliggande moduler uppdateras

Paketbeskrivning: package.json

```
{
  "name": "thin-red-line",
  "version": "0.5.7",
  "private": true,
  "scripts": {
    "start": "grunt docs && node ./bin/www",
    "test": "mocha --recursive test/unit_test -R dot",
    "system-test": "jasmine-node test/system_test"
  },
  "dependencies": {
    "bluebird": "~3.0.6",
    "body-parser": "~1.10.2",
    "xml2js": "^0.4.9"
  },
  "devDependencies": {
    "chai": "^3.0.0",
    "chai-as-promised": "5.1.*",
    "rewire": "~2.5.1"
  }
}
```

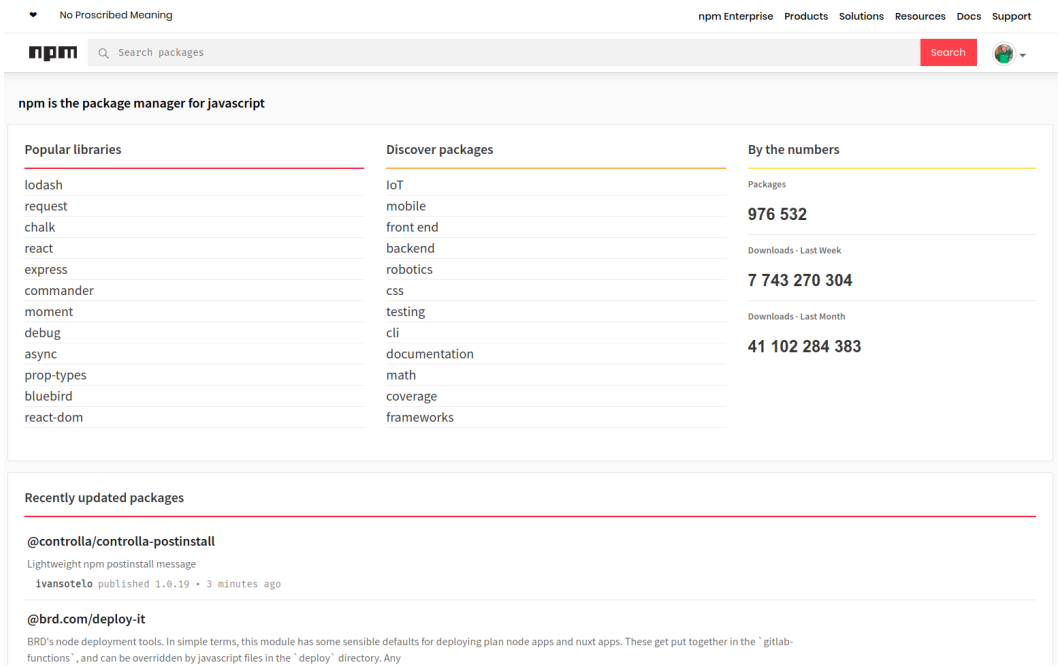
MALMÖ UNIVERSITET

Demo

Johan leker med npm

MALMÖ UNIVERSITET

npmjs.com

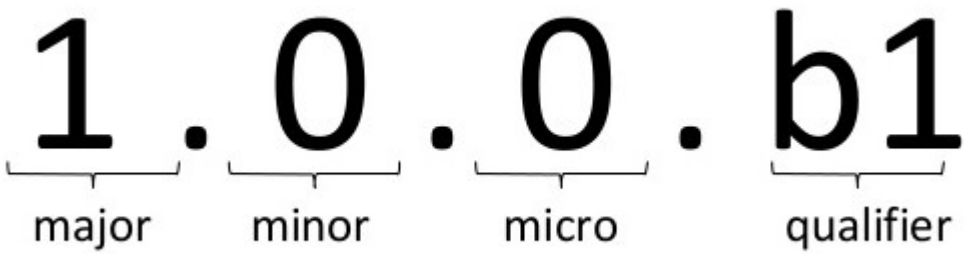


MALMÖ UNIVERSITET

npmjs.com

- Npm:s repository för JavaScript-moduler
- Modulerna är fria att använda i dina projekt
- Använder **semantisk versionering** för att skilja mellan olika versioner av moduler

Semantisk versionering



Semantisk versionering: http://semver.org/

MALMÖ UNIVERSITET

Paketering

MALMÖ UNIVERSITET

Vårt problem

- Riktiga mjukvaruprojekt är stora
- Riktiga mjukvaruprojekt behöver paketeras innan de levereras
- Hur hanterar vi detta på ett effektivt sätt?

Vad är paketering?

Utvecklad mjukvara innehåller som regel kod och artefakter som är ointressanta för slutanvändaren. Dessa tar stor plats och kan dessutom vara säkerhetshål. En paketerare tar bort alla onödiga artefakter och levererar en färdig mjukvara.

Begreppen “build” och “bundle”

- En build är den process som tar din utvecklingskod – källkod, grafiska element och annat – och slår samman dem till en enhet.
- En bundle är den artefakt som är resultatet av din build-process.

Paketering med Webpack

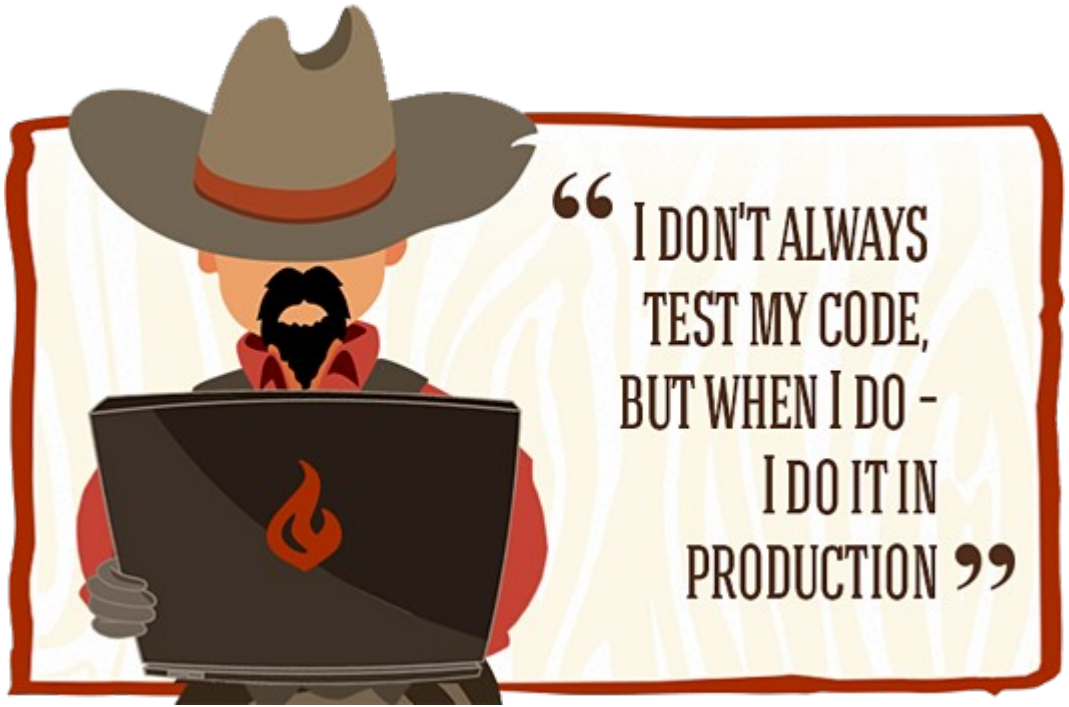
- Webpack är en av många – men en av de mer användna – paketeringsverktygen för webbprojekt
- Skrivet i JavaScript
- Slår samman flera filer av samma typ till en (se demo)
  - Flera JavaScriptfiler blir en
  - Flera CSS-dokument blir ett, etc
- Genererar en uppsättning filer per HTML-dokument som ska användas
  - Sparar mycket laddningstid
- Klarar i grunden av JavaScript, men kan utökas med andra filtyper
- Hanteras med hjälp av konfigurationsfiler – precis som pakethanterarna

Demo

Johan leker med Webpack

MALMÖ UNIVERSITET

Testning



MALMÖ UNIVERSITET

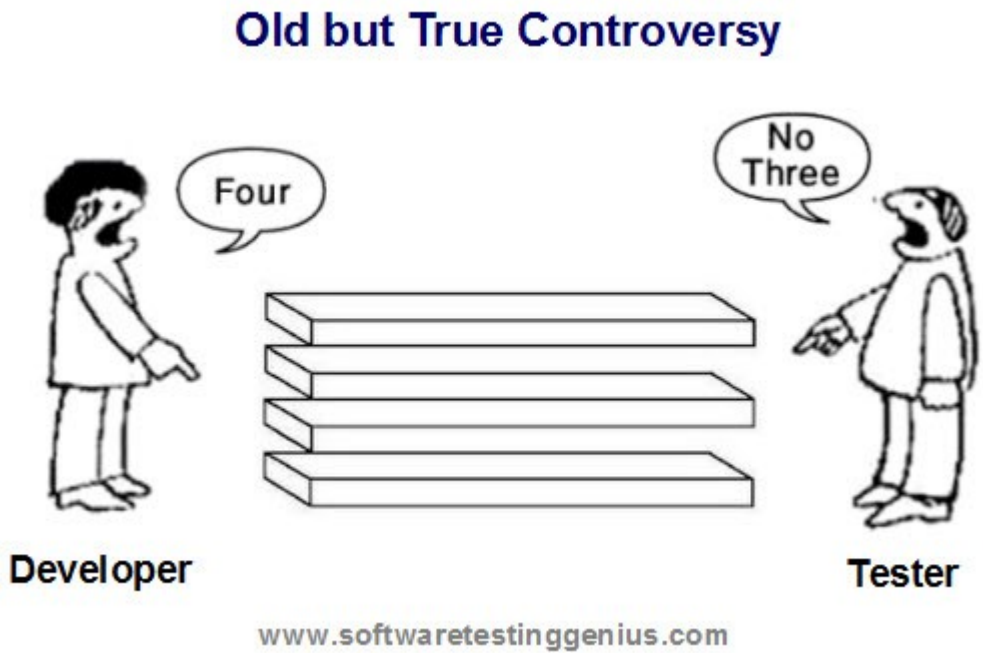
Vad är testning?

> **“Software testing** is an investigation conducted to provide stakeholders with information about the quality of the product or service under test.”
>
> Enl. Wikipedia

- Validering → Bygger vi rätt saker?
- Verifikation → Bygger vi sakerna på rätt sätt?

Wikipedia, *Software testing* – https://en.wikipedia.org/wiki/Software_testing

Varför testar vi våra applikationer?



MALMÖ UNIVERSITET

Varför testar vi våra applikationer?

- Tekniska skäl
- Utvecklingsteamets skäl
- Ekonomiska skäl

MALMÖ UNIVERSITET

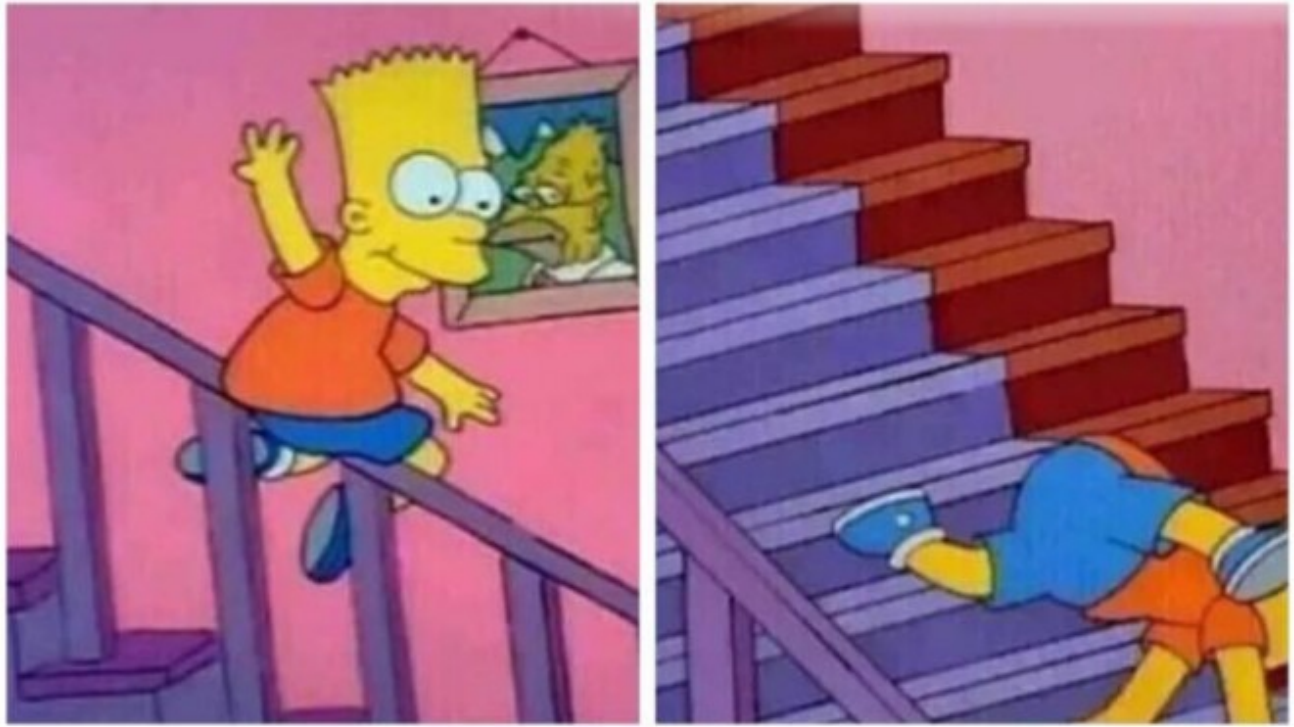
Tekniska skäl: Säkerställ funktionaliteten

Det här är validering och verifiering!

- Fungerar koden som den är tänkt att göra?
  - Hanteras indatan på rätt sätt?
  - Fungerar koden med felaktig indata?
  - Är koden feltolerant?
- Kommer programmet att fungera i produktion?
  - Matchar vår utvecklingsmiljö produktionsmiljön?
  - Fungerar all kod tillsammans?

My code working well on on my machine

* *Deploys* *



MALMÖ UNIVERSITET

Utvecklingsteamets skäl: Förtroende

- En bra utvecklare kan visa att hens kod fungerar
- Bra tester säkerställer att koden fungerar
  - Tester är bra att ha under utvecklingen av en funktion
  - De är ännu bättre att ha när koden har levt en tid
- Tester kan användas som bas i diskussioner

Utvecklingsteamets skäl: Historik och nya utvecklare

Tester är dokumentation → förenklar introduktion av nya utvecklare

- Väl utformade och beskrivna tester visar hur en klass eller funktion ska fungera
- BDD-tester (user stories-baserade tester) beskriver hur applikationen ska fungera
- Lösta buggar och fel visas med tester

MALMÖ UNIVERSITET

Ekonomiska skäl:
Driftstörningar är dyrare än utvecklingstid

- Mjukvarutestning är dyrt
  - Längre utvecklingstid
  - Fler utvecklare/testare
  - Mer infrastruktur
- Fel i mjukvaran är dyrare
  - Nertid (se nästa slide)
  - Dålig PR/goodwill

MALMÖ UNIVERSITET

Ekonomiska skäl:
Driftstörningar är dyrare än utvecklingstid

Average Cost of Downtime

Even if you company survives a disaster, the costs are staggering:

- Brokerage $6M - $7M / hour
- Banking $5 - $6M / hour
- Credit Card $2M - $3M / hour
- Pay Per View $1 - $2M / hour (up to $50M for fights)
- Airline Reservations $1M / hour
- Home Shopping $100K / hour
- Catalog Sales $100K / hour
- Tele-ticket $70K / hour
- Package Shipping $30K / hour
- ATM Fees $20K / hour

Average mean time to repair or recover: 4.0 hours

Morpheus, *How to manage app uptime like a boss* - https://www.morpheusdata.com/blog/2016-04-06-how-to-manage-app-uptime-like-a-boss

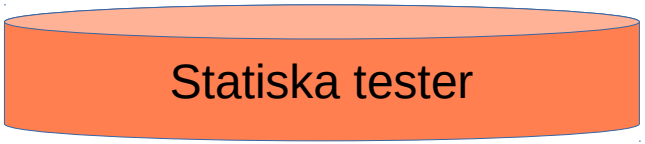
MALMÖ UNIVERSITET

Olika typer av testning

- Statisk testning
- Enhetstestning
- Integrationstestning
- End-to-end/acceptanstest

Statisk testning

- Kräver ingen körning av mjukvaran
- Innefattar analys av:
  - Krav
  - Designdokument
  - Koden
- Kan göras både manuellt och med verktyg



Enhetstestning

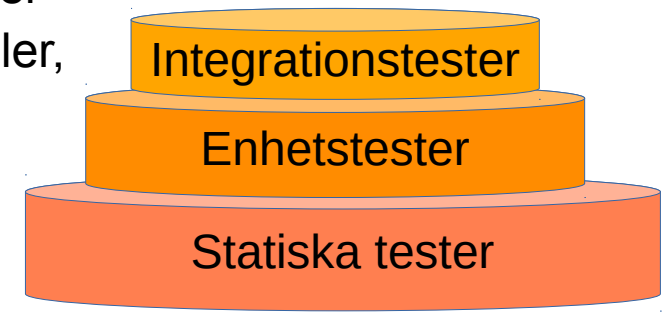
- Testning av små delar – enheter – av koden, exempelvis klasser eller funktioner
- Körs ofta – så fort koden ändras
- Testfallen bör definieras innan koden skrivs

- Utförs med verktyg, oftast automatiskt
- Kan ses som specifikation och dokumentation

MALMÖ UNIVERSITET

Integrationstestning

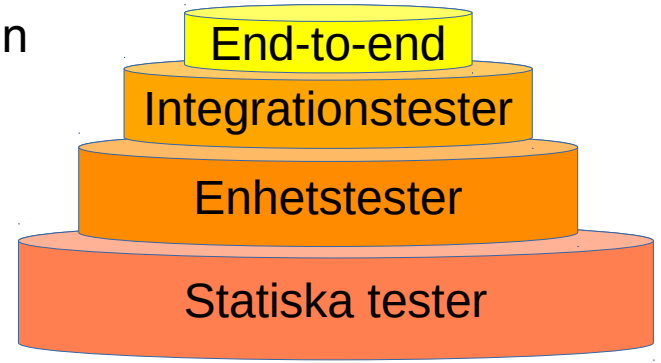
- Tester som involverar flera enheter av koden – exempelvis två moduler, klasser eller funktioner
- Görs för att säkerställa att enheterna fungerar tillsammans



- Testerna körs ofta som en del av förberedelserna inför en release – projekt tenderar att använda specifika verktyg för detta

End-to-end-testning

- Testar användarflöden i mjukvaran
- Heltäckande och testar många funktionaliteter samtidigt
- Körs inför leveranser
- Görs påfallande ofta av vanliga människor, men kan även göras med hjälp av automatiserade verktyg



MALMÖ UNIVERSITET

Demo

Johan leker med Jest

MALMÖ UNIVERSITET

Till sist

MALMÖ UNIVERSITET

Glöm inte att programmering är kul!

Outcognito Mode

Outcognito Mode is a Chrome extension that publicly tweets every website you visit and everything you type. By Harold Cooper.

Outcognito Mode @outcognitomode · Feb 25
I just typed pancakes

Outcognito Mode @outcognitomode · Feb 25
I just went to allrecipes.com

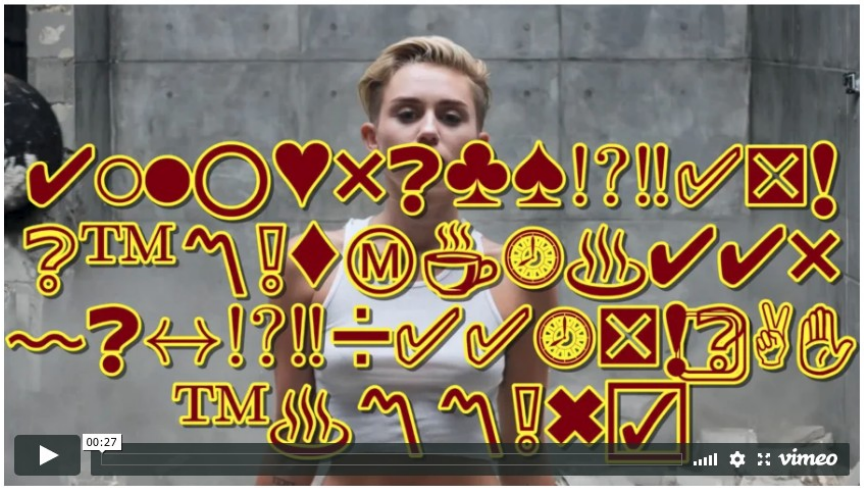
Outcognito Mode @outcognitomode · Feb 25
I just typed delicious food

Outcognito Mode @outcognitomode · Feb 25
I just went to

Indian Saag Recipe
Indian saag is a curry of cooked mustard greens and spinach, tasty with naan bread.
allrecipes.com

The Emoji Subtitle Creator

The Emoji Subtitle Creator by Ross Goodwin and Seth Kranzler automatically translates normal subtitles into ascii symbols.



MALMÖ UNIVERSITET

Det är okej att vara cowboy ibland

MALMÖ UNIVERSITET