Låt datorn göra skitjobbet

Gulp och Webpack

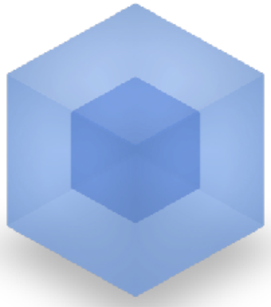
Diskutera vanliga handlingar som kan automatiseras.

Varför, varför inte, ansvar för vem?

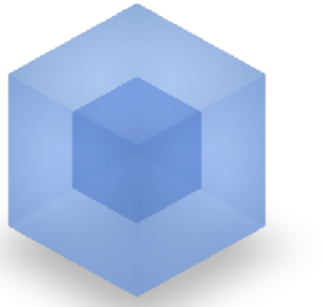
Verktygslådan

Package managers

Task runners



Module bundlers



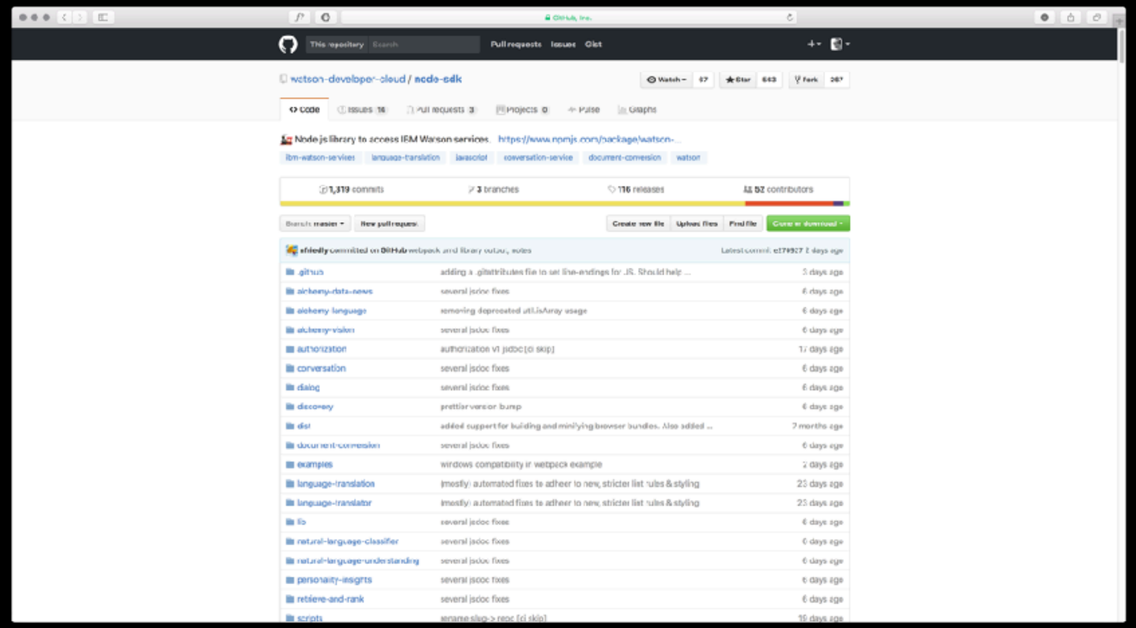
webpack
MODULE BUNDLER

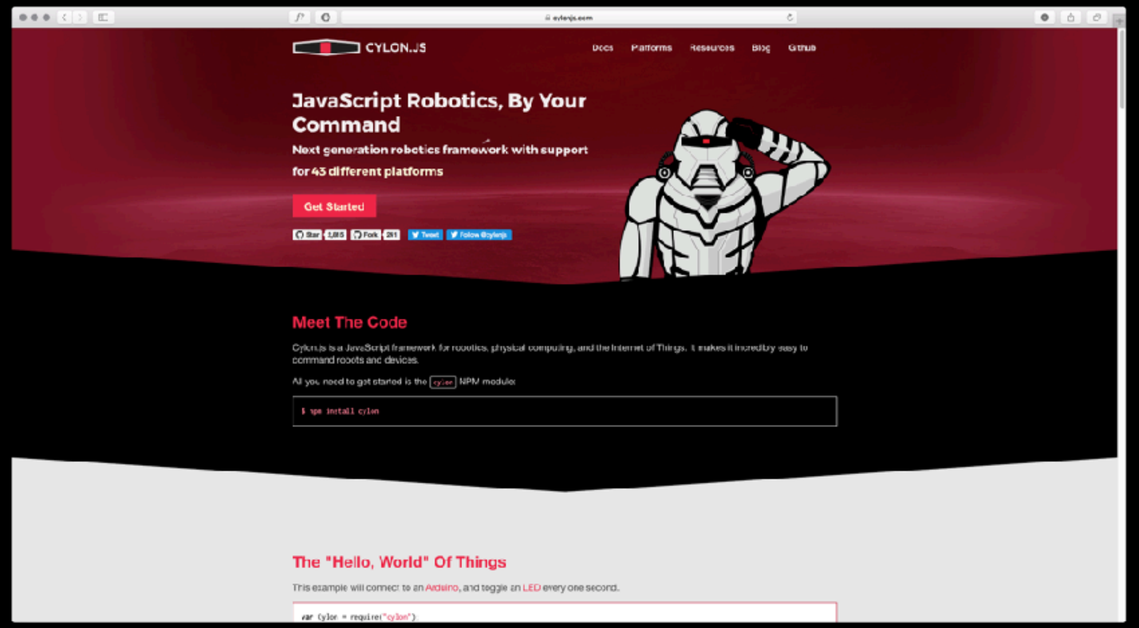
rollup.js
the next-generation JavaScript module bundler

Node driver alla (?) dessa verktyg

Baserat på Googles V8-motor (JS)

Node används i massa saker





Demo: Node

Global/lokal installation

- npm install [package] -g
  - Global installation finns i hela ditt filsystem
  - Vissa moduler kräver detta för att fungera
  - Vanligtvis vill du inte ha globala installationer

Dependencies

- npm install [package]
  - Dependencies (—save, —S)
  - Developer dependencies (—save-dev, —D)

Package.json

- Innehåller alla dependencies...
- ...samt information om version (semver) och upphovsperson etc.
- Kan manuellt behandlas, men man bör inte göra det
- Skall alltid vara synkad mellan alla som jobbar i projektet

Mission critical

- Komplexa saker som npm är beroende av en ganska fin balans mellan olika projekts relationer och inter-relationer.
- Oväntade förändringar kan potentiellt förstöra projekt som inte uppenbart har en relation till det som ändrats.
- ^[vers.nr] — godkänn alla förändringar utom nytt huvudnummer
- ~[vers.nr] — godkänn endast buggfixar

Shrinkwrap

- Låser ner alla versionsnummer, även på djup nivå
- Skapar en egen package.json (som inte ersätter den vanliga)
- Shrinkwrap-paketet gäller före vanliga package.json när andra installerar via npm

Gulp

Gulp har fyra metoder

- task
- watch
- src
- dest

Gulp har fyra metoder

- task — utför
- watch — utför om något händer med det som “watchas”
- src — ange källa
- dest — ange slutmål

Strömmande arkitektur

- Gulp bygger på streaming, eller stream piping
- Detta innebär att en ström förflyttar sig mellan de olika behandlingar (tasks) som utförs på den
- Eventuella fel får hela strömmen att kvävas
- Tasks är inte som standard synkrona och måste inte följa en viss ordning
- Denna modifierade ström kommer slutligen ut som en fil
- Till skillnad mot Grunt så mellanlagras inte denna ström som en annan fil

Gotchas

- Ingen undo: lätt att paja saker som inte kan återställas
- För nybörjare är felmeddelandena sällan särskilt informativa
- Nästan ingen funktionalitet följer med från start
- Systemet har ofta ett stort behov av andra moduler

Packages

- Gulp-moduler är Node-moduler
- Ibland är Node-moduler “wrappade” för att bättre funka med streaming (exempelvis har Webpack en sådan Gulp-wrapper)
- Du kan vanligtvis köra både Node-moduler och Gulp-moduler, men givetvis inte ex. Grunt-moduler

Demo: Colorguard

Demo: Imagemin

“Lågnivåpaket”

- del
- rename
- reporter
- changed
- size
- util

Vanliga packages

- requireDir
- gulp-concat
- gulp-sass
- gulp-uncss
- gulp-browser-sync
- gulp-babel

Globbing

Ett mönster för att “smart” göra ett urval

Globbing

```
babel: {
      src: [dev + "scripts/js/**/*",
         "!" + dev + "scripts/js/_unused/**/*",
         "!" + dev + "scripts/js/nocompile/**/*",
         "!" + dev + "scripts/js/init/**/*"
          ],
      dest: dev + "scripts/compiled/"
}
```

Config-objekt

Ett config-objekt centraliserar variabler och referenser vilket hjälper i komplexa Gulp-installationer

Config-objekt

```
const dev = "Static/";


module.exports = {
    browsersync: {
        notify: true,
        server: {
            baseDir: dev,
            online: false,
            notify: true,
            injectChanges: true
        }
    }
}
```

Visual Studio: Task Runner Explorer

- Funkar utmärkt även med komplexa Gulp-installationer!
- Ingen särskild konfiguration krävs, förutom...
- ... att du manuellt måste peka om VS till en egen Node-installation

Webpack

Entries

- Webpack letar efter moduler du länkar in via entry-filer
- Du kan ange en eller flera entry-filer

Vendor bundling

- Ett vanligt use case är att bundla vendor-filer i en separat fil
- Detta är precis som vilken annan entry som helst...
- ...med skillnaden att man ibland vill använda en plugin som VendorChunkPlugin

Chunking

- Code splitting
- “On demand loaded chunk"
- Alla dependencies splittas i flera filer

Hashing

- Gör att vi kan cache-busta hos klienten
- Måste som vanligt refereras till via exakt namn i HTML
- Kombineras ofta med Webpack-plugin för att rendera HTML

Tree-shaking

- Webpack kontrollerar rekursivt alla filer för att ta bort “död kod”
- Introducerades i nya Webpack 2
- Har potential att göra kod mycket mindre och snabbare

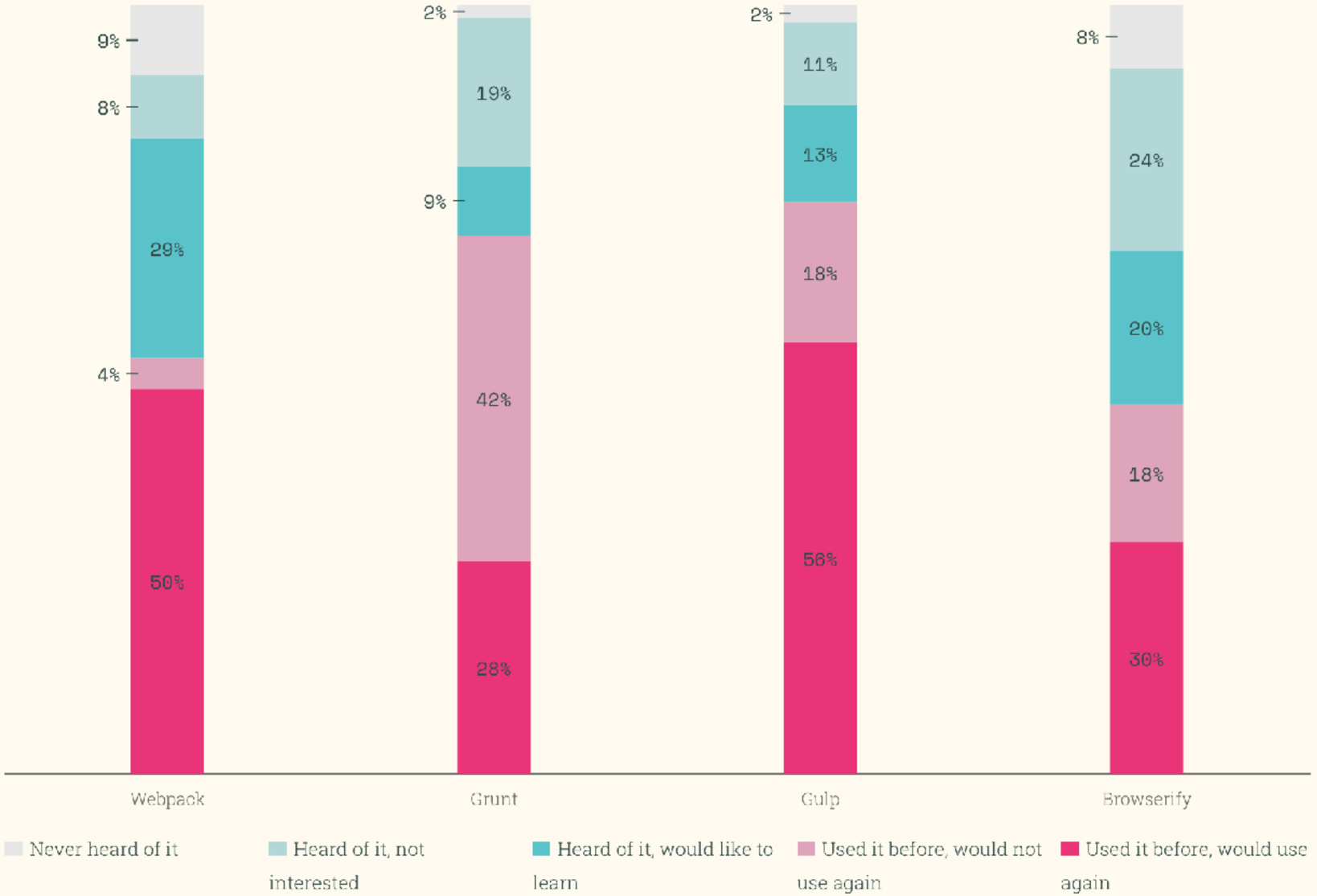
Framtiden för byggverktyg

Kommer bli helt standard inom mycket kort tid

Build Tools

Filter: All | Interest | Satisfaction

Show: Percents | Numbers



“Unlike the other sections of the survey which have at least one or two options people haven’t heard of, practically every developer surveyed was at least aware of the existence of the four survey options, which all clocked in at 91% awareness or higher.”

- http://stateofjs.com/2016/buildtools/

Hierarkier och systemägande: Arkitekt eller ledutvecklare bestämmer byggprocessen likt i devops/backend

Tekniken kommer sannolikt stabilisera sig något under 2017, då dessa verktyg utvecklats under en mycket kort tid för ett delvis nytt område

Workshop

Skapa en Gulp-task.

Skapa och använd ett config-objekt.

Skapa en sekvens av tasks.

— Testa både med och utan runSequence.
— Prova olika ordning (synk/asynk).

Bundla filer med Webpack.

Diskutera vilka grundläggande behov man kan lösa i kommande projekt genom att tidigt forma ett byggsystem.

Börja bygga ett mer utförligt byggsystem för några av de problem och behov ni identifierat.