NODE, NPM & BYGGVERKTYG

VAD ÄR NODE

Node.js® *is a JavaScript runtime built on Chrome's V8 JavaScript engine. Node.js uses an* ***event-driven, non-blocking I/O model*** *that makes it lightweight and efficient. Node.js' package ecosystem, npm, is the largest ecosystem of open source libraries in the world.*

Ref: Node.js

SIMPLIFIED

Node gör vad din webbläsare kan göra med

JavaScript *fast utan din webbläsare*

Köra JavaScript från **terminalen**

Inget visuellt, html/css

Bara JavaScript

Ingen DOM

Ingen `index.html` som kopplar samman våra filer

INTE BARA BACKEND

Node kan användas för att skapa servrar men det är långt från allt

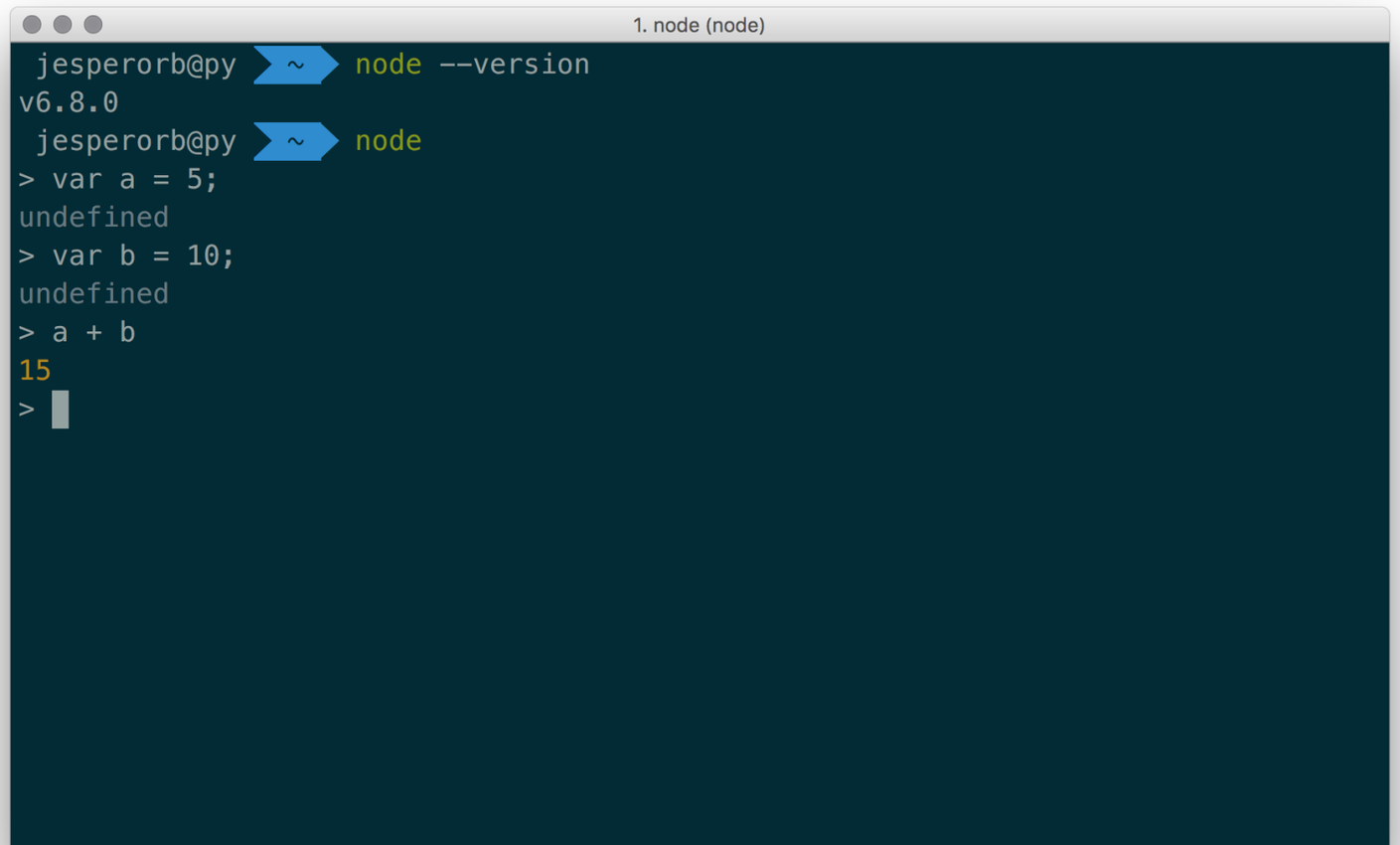
JavaScript är ett språk som alla andra. **npm** är till för att ladda ner och hantera moduler, modulerna behöver dock inte vara ens JavaScript.

Vi måste inte heller göra massa backend-grejer, vi kan använda andra verktyg genom npm

COMMAND LINE INTERFACE

När ett program har ett grafiskt gränssnitt snackar man om GUI
(Graphical User Interface)

Vi använder oss utav flera olika program som körs från terminalen: CLI
(Command Line Interface)


1. node (node)
jesperorb@py ~ node --version
v6.8.0
jesperorb@py ~ node
> var a = 5;
undefined
> var b = 10;
undefined
> a + b
15
>

```
//main.js
console.log("Hej!");
```

```
node main.js
# Hej!
```

Vilken version spelar inte så stor roll : **6/7** är senaste

Ni kommer få problem oavsett version. Paket som krånglar etc.

Open source ecosystem : privatpersoner som underhåller en stor del av biblioteken gratis

REQUIRE();

Istället för att länka samman allt i **index.html** importerar man kod med node-syntax

```
var code = require("code.js");
```

Vi importerar oftast färdiga moduler/paket

MÅSTE VI LÄRA OSS NODE?

Nej, det kan vara bra att kunna grunden men för det mesta använder vi oss av node p.g.a. Node Package Manager för att importera kod till våra projekt

Standard för JavaScript-utvecklare att använda detta ekosystem för utveckling.

EXEMPEL PÅ NODE-SERVER

```
//Minimum to create a node-server
var http = require("http");
var server = http.createServer(function(request, response) {
  response.send('Hello');
  response.end();
});
server.listen(4000);
```

```
//Minimum to create a node-server
var http = require("http");
var server = http.createServer(function(request, response) {
  //We can also send files and read files
  response.sendFile('index.html');
  response.end();
});
server.listen(4000);
```

NPM

npm is the package manager for JavaScript.

N ode P ackage M anager

PACKAGE/MODULE/LIBRARY

Färdigpaketerad kod som har en specifik uppgift att utföra

Ett paket kan t.ex. vara en funktion.

Moduler bygger på andra moduler: dependencies

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

Code pulled from NPM – which everyone was using

Careful, careful ... Don't fumble this like the JS world (Credit: Claus Rebler)

freak scenario

```
function leftpad (str, len, ch) {
  str = String(str);
  var i = -1;
  if (!ch && ch !== 0) ch = ' ';
  len = len - str.length;
  while (++i < len) {
    str = ch + str;
  }
  return str;
}
```

MODULER

node-moduler är verktyg som ska underlätta vårt kodande.

Saker som autoprefixing, minification och konvertering av sass kan allt göras via node-moduler.

Och allting körs via terminalen, via node

NPM INIT

BYGGVERKTYG

BYGGVERKTYG

Verktyg som ska hjälpa oss under utvecklingsprocessen

Sass är ett sådant verktyg kan man säga

sass filer använd bara under utvecklingen, vi konverterar alltid till `css`

```
sass sass:css --watch
```

SASS

Hur installerade vid sass?

Installerade **ruby** en körmiljö för språket ruby

Med ruby medföljer pakethanteraren RubyGems

```
gem install sass
```

NODE-SASS

node har en port av sass också

```
npm install node-sass
```

FLERA VERKTYG?

Minifera filer t.ex

Ladda om webbläsaren vid uppdatering av css/js

OM VI SKA KÖRA FLERA SAKER?

Vi vill konvertera sass, minifera och autoprefixa våra filer

```
sass scss:css && minifier main.js main.min.js
```

börjar bli rörigt och tänk om vi ska göra 10 saker till?

Alternativt lägga i **npm start**

GULP

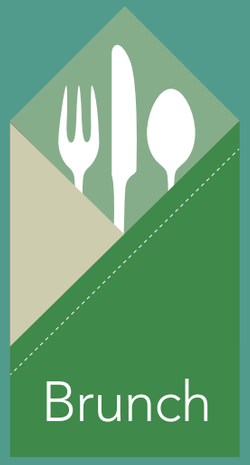
GULP

Gulp är en **Task Runner** som är skapat för att lösa vårt tidigare problem: automatisera uppgifter.

ett **CLI** som installeras via npm och körs genom node


Gulp

GRUNT



SKILLNADER

Massa skillnader men huvuduppgiften är densamma, paketera dina filer och automatisera.

Alla verktyg behöver alltid en **config-fil**

Vissa verktyg framhäver att de behöver mindre config.

Gulp är ett verktyg jag upplever behöver lite config och krånglar sällan på lättare uppgifter.

EXEMPEL

GULP-FUNKTIONER

- `gulp.task()` : uppgiften som ska köras
- `gulp.watch()` : som `--watch` vi kollar efter förändringar
- `gulp.src()` : våra ursprungliga filer som ska konverteras
- `gulp.dest()` : var våra färdiga filer ska hamna

```
gulp
```

Om vi bara skriver **gulp** så kör gulp sin 'default' -task.

Har vi t.ex. en task som heter **sass** kan vi köra den såhär:

```
gulp sass
```

TASKS LÄNKADE TILL VARANDRA

```
gulp.task('default', ['sass'], ()=>{
  //sass task runs when this task runs
}
```

NODE_MODULES

Alla moduler som ditt projekt använder måste vara installerade i **node_modules**

```
npm install   //Install all in package.json
```

```
npm install -g gulp   //Install globally
```

```
npm install --save-dev gulp //Install locally
```

```
npm install --save gulp //Also install locally
```

GÖR ALLTID

`npm init`

`npm install --save-dev package_name`

Ditt projekt måste ha en `package.json`

När du installerat en modul ska du ha en mapp som heter `node_modules`

INGEN DIREKT UPPGIFT

06_node.md innehåller mer info om node/npm och vilka steg man ska ta vid användande av npm

gulp-boilerplate -repot innehåller en färdig config som ni kan använda.

Se till så att ni har fått det att fungera.