Architektur von Informationssystemen

Hochschule für angewandte Wissenschaften

Sommersemester 2016

Nils Löwe / nils@loewe.io / @NilsLoewe

4. Praktikum

Praktikum 4: Architekturentwurf

Fragen?

Wiederholung

Was ist Softwarearchitektur?

Geschichte und Trends

Sichten auf Architekturen

Qualiät und andere nichtfunktionale Anforderungen

Architekturmuster

Dokumentation von Architekturen

Technologien und Frameworks

Ruby on Rails

Spring Boot (moovel Group GmbH)

NodeJS

AngularJS

Docker (Akra GmbH)

AMQP

Twitter Bootstrap

Microservices (Wer liefert was GmbH)

Spring Boot (moovel Group GmbH)

http://blog.jan-ahrens.eu/spring-boot-intro/#/

Spring Boot (moovel Group GmbH)

DEPENDENCY INJECTION (DI)

Also known as inversion of control (IoC)

Hollywood principle: ”Don’t call us, we call you”

Spring Boot (moovel Group GmbH)

BENEFITS OF DI

Modularization

Testable code

Decrease coupling

Behavior can be changed

Spring Boot (moovel Group GmbH)

ASPECT ORIENTED PROGRAMMING (AOP)

Separate cross-cutting concerns

In plain English: reduce duplications

Aspect = concern

Spring Boot (moovel Group GmbH)

EXAMPLES FOR CROSS-CUTTING CONCERNS

Logging

Transaction management

Authentication

Spring Boot (moovel Group GmbH)

THE SPRING FRAMEWORK

Convention over configuration

Easily extensible through ”Starter POMs”

Micro-framework paradigm

FURTHER READING

http://start.spring.io

http://spring.io/guides

http://docs.spring.io/spring-boot/docs/current/reference/html

http://ratpack.io

Ruby on Rails

Spring Boot (moovel Group GmbH)

NodeJS

AngularJS

Docker (Akra GmbH)

AMQP

Twitter Bootstrap

Microservices (Wer liefert was GmbH)

Node.js

Node.js

Node.js ist eine serverseitige Plattform für Netzwerk-Anwendungen, die auf der Google Chrome JavaScript Engine (V8 Engine) basiert.

- easily build fast and scalable network applications
- event-driven, non-blocking I/O model
- lightweight, efficient, perfect for data-intensive real-time applications
- large JavaScript Library

Node.js

Node.js = Runtime Environment + JavaScript Library

https://github.com/nodejs/node

Node.js

Releases

- Current: Released from active development branches of this repository
- LTS: Releases that receive Long-term Support, with a focus on stability and security
- Nightly: Versions of code in this repository on the current Current branch

Node.js

Chrome V8

Google's high performance, open source, JavaScript engine.

- Open source, high-performance JavaScript engine
- Written in C++
- Used in Google Chrome
- It implements ECMAScript as specified in ECMA-262, 3rd edition
- Runs on Windows XP or later, Mac OS X 10.5+, and Linux systems that use IA-32, ARM or MIPS processors
- V8 can run standalone, or can be embedded into any C++ application

Node.js

Anwendungsfälle

- I/O bound Applications
- Data Streaming Applications
- Data Intensive Real time Applications (DIRT)
- JSON APIs based Applications
- Single Page Applications

Node.js

Grenzen

- CPU intensive applications

Node.js

REPL: Read Eval Print Loop

- Read - Reads user's input, parse the input into JavaScript data-structure and stores in memory.
- Eval - Takes and evaluates the data structure
- Print - Prints the result
- Loop - Loops the above command until user press ctrl-c twice.

REPL: Read Eval Print Loop

- REPL feature of Node is very useful in experimenting with Node.js codes and to debug JavaScript codes.
- REPL can be started by simply running node on shell/console without any argument as follows.

```
$ node
>
```

Node.js: NPM

Node Package Manager

- https://www.npmjs.com/
- Find, share, and reuse packages of code from hundreds of thousands of developers
- 3 million developers and thousands of companies use npm

Node.js: NPM

Statistiken (Stand 13.05.2016)

- 283,432 total packages
- 57,724,266 downloads in the last day
- 1,040,776,238 downloads in the last week
- 4,169,158,104 downloads in the last month

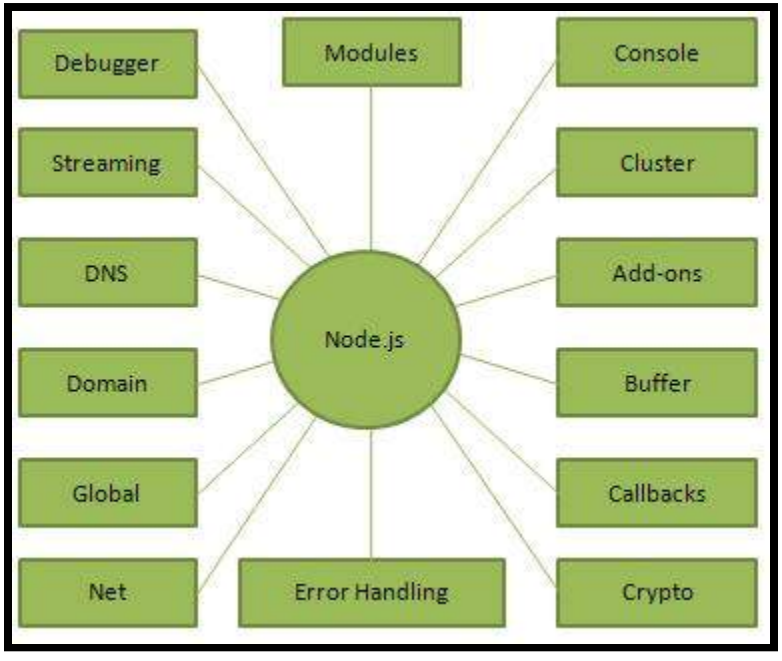
Node.js: NPM

Geschäftsmodell

- Open Source: FREE
- Private Account: $7 per user / month
- Orgs: $16 per user / month
- Enterprise: $2000 per year
- npm Enterprise Pro

Node.js

Concepts



Node.js

Callbacks

- Asynchronous equivalent for a function
- A callback function is called at the completion of a given task
- All APIs of Node are written is such a way that they supports callbacks.

Node.js

Callbacks: Blocking Example

main.js

```
var fs = require("fs");

var data = fs.readFileSync('input.txt');

console.log(data.toString());
console.log("Program Ended");
```

input.txt

```
This is the input.txt file
```

```
$ node main.js
This is the input.txt file
Program Ended
```

Node.js

Callbacks: Non-Blocking Example

main.js

```
var fs = require("fs");

fs.readFile('input.txt', function (err, data) {
if (err) return console.error(err);
console.log(data.toString());
});

console.log("Program Ended");
```

input.txt

```
This is the input.txt file
```

```
$ node main.js
Program Ended
This is the input.txt file
```

Node.js

Event Driven Programming

- The functions which listens to events act as Observers
- When an event gets fired, its listener functions start executing
- Node.js has multiple in-built events available through the **events module** and **EventEmitter class**

Node.js

Streams

Streams are objects that let you read data from a source or write data to a destination in continous fashion.

Node.js

Streams

In Node.js, there are four types of streams.

- Readable - used for read operation.
- Writable - used for write operation.
- Duplex - can be used for both read and write operation.
- Transform - A type of duplex stream where the output is computed based on input.

Node.js

Streams

Each type of Stream is an EventEmitter instance and throws several events

- data - fired when there is data is available to read.
- end - fired when there is no more data to read.
- error - fired when there is any error receiving or writing data.
- finish - fired when all data has been flushed to underlying system

Node.js

File System

- Node implements File I/O using simple wrappers around standard POSIX functions
- Every method in fs module have synchronous as well as asynchronous form
- Asynchronous methods take a parameter as "completion function callback"
- It is preferred to use asynchronous methods instead of synchronous methods

Node.js

Standard Modules

- OS Module: Provides basic operating-system related utility functions.
- Path Module: Utilities for handling and transforming file paths.
- Net Module: Servers and clients as streams
- DNS Module: DNS lookup, operating system name resolution functionalities
- Domain Module: I/O operations as a single group.
- http module: Create either HTTP client of server

Node.js

Express

- Minimal and flexible Node.js web application framework
- Robust set of features to develop web and mobile applications
- Allows to set up middlewares to respond to HTTP Requests
- Defines a routing table for actions based on HTTP Method and URL
- Allows to dynamically render HTML Pages based on passing arguments to templates

Node.js

JXcore

- Compile and distribute it a Node.js app
- Open source project
- Packaging and encryption of source files and other assets

Fragen?

Ruby on Rails

Spring Boot (moovel Group GmbH)

NodeJS

AngularJS

Docker (Akra GmbH)

AMQP

Twitter Bootstrap

Microservices (Wer liefert was GmbH)

AngularJS

AngularJS ist ein clientseitiges JavaScript-Webframework für Single-page-Webanwendungen nach dem Model-View-ViewModel-Muster

AngularJS

- Clientseitige Generierung von HTML-Seiten
- Funktionalität ohne DOM-Manipulation via jQuery
- Datenvalidierung von Eingabeformularen als Funktionalität

AngularJS

Komponenten

Die Strukturierung eines Angular-Webclients erfolgt auf Basis von

- Modulen
- View-Templates
- Controllern
- Scopes
- Filtern
- Providern (Factory, Service)

AngularJS

Datenbindung

- Datenbindung nach dem MVVM-Muster: Einfache Synchronisation zwischen View und Anwendungslogik
- Deklarative Beschreibungen von Datenbindungen innerhalb der View
- Event-Schleife fängt jede Eingabe ab und aktualisiert ggf. Teile der View
- Nicht editierbare Daten können mittels One-Time-Binding von weiteren Aktualisierungen ausgeschlossen werden.

AngularJS

Mocking-Modul

- Standardfunktionalitäten wie $http können für Tests ersetzt werden
- Umsetzung von isolierten Testfällen

AngularJS

Controller

- Clientseitiges Model
- Controller werden zu einem Modul zusammengefasst
- Module werden mittels eines Dependency-Injection-Containers in die Applikation eingebunden
- -> Dabei wird die View mit dem Model verbunden
- -> Bidirektionale Datenbindung

AngularJS

Direktiven

- benutzerdefinierte HTML-Elemente und -Attribute
- Vordefinierte Direktiven sind am ng-Namensraum erkennbar
- Um Elemente auszuwählen, verwendet AngularJS ein integriertes jQuery Lite (jqLite)
- Wird jQuery in das HTML-DOM eingebunden, wird dieses statt jQuery Lite verwendet.

AngularJS

Interpolation

- Mit doppelten geschwungenen Klammern können JavaScript-Ausdrücke ins HTML eingebettet werden
- Mit dem Pipe-Operator | können Filter hinzugefügt werden

AngularJS

Services

- enthalten die Geschäftslogik
- binden externe Ressourcen wie z.B. REST-Webservices ein
- werden als Singleton instanziiert
- können selbst implementiert werden oder von Dritten eingebunden werden (Beispiele $http und $resource)

AngularJS

Routen

- Zuordnung von URLs zu Views mittels *ngRoute*
- Unterschiedliche HTML Seiten können mittels *ng-view* nachgeladen werden.
- Eine *ng-view* Direktive pro Seite
- Das *$location* Objekt erlaubt die direkte Verarbeitung der Browser-URL

AngularJS

Vorteile

- Testbarkeit
- Refaktorisierbarkeit
- Wiederverwendbarkeit
- Verwendung das Javascript Datentypen

AngularJS

Nachteile

- Erstmal nur clientseitig
- Breaking Changes von V1 auf V2
- Performance
- Relativ komplex -> Viele Möglichkeiten Dinge falsch zu machen

AngularJS

Minimales Beispiel

Angular minimales Beispiel

AngularJS

Komplexeres Beispiel

- TodoList Beispiel
- Beispiel Code [github]

AngularJS

Quellen

- https://angularjs.org
- https://github.com/angular/angular.js
- https://docs.angularjs.org/guide

Fragen?

Vorbereitung auf Klausuraufgaben

- Was ist Dependency Injection?
- Nennen Sie drei Beispiele für Crosscutting Concerns
- Wo wird NodeJS typischerweise eingesetzt?
- Was ist das grundlegende Konzept einer NodeJS Anwendung?
- Wo sollte NodeJS eher nicht eingesetzt werden?
- Für welche Art von Anwendungen wird AngularJS verwendet?

Fragen?

Unterlagen: ai2016.nils-loewe.de