FH JOANNEUM
University of Applied Sciences

Toolchain

John Feiner

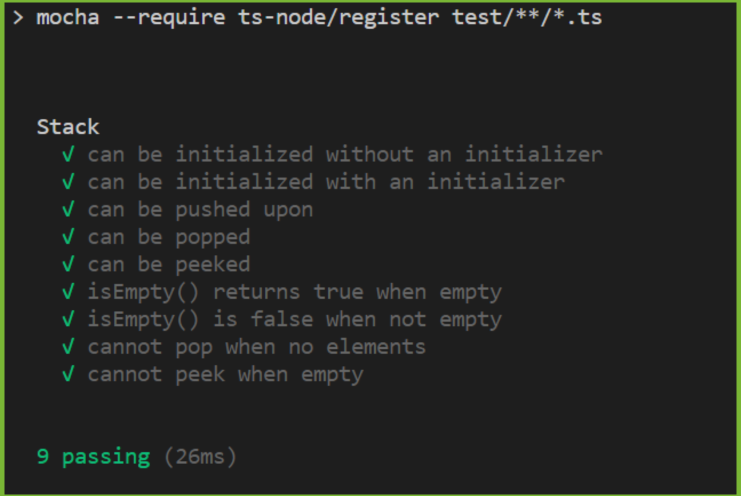
Tool Chain

Automate, automate, automate, automate, ...

Command line: e.g. provide "setup.sh"
(optional: IDE)

Configurations (never IN code): IP, Port, Username, Token

Development & Testing



```
> mocha --require ts-node/register test/**/*.ts



  Stack
    √ can be initialized without an initializer
    √ can be initialized with an initializer
    √ can be pushed upon
    √ can be popped
    √ can be peeked
    √ isEmpty() returns true when empty
    √ isEmpty() is false when not empty
    √ cannot pop when no elements
    √ cannot peek when empty


  9 passing (26ms)
```

https://medium.com/@FizzyInTheHall/run-typescript-mocha-tests-in-visual-studio-code-58e62a173575

John Feiner

Coding & Compiling & Debugging

Project configurations, include libs:

package managers (npm, yarn, Webpack)

https://yarnpkg.com

TypeScript compiling (tsconfig.json, tsc)

Teams, GIT

Logging (Poor Man's Debugging)

Debugging (on device)

Testing (Test Suites)

Unit tests (asynchron)

UI tests (remote, on real devices)

Performance tests

https://jestjs.io

Again: please automate: jest ..

Running Code Metrics

Quality

- Style Guides

Security

- Security Metrics

Again: please automate: *tshint* ..

Run Automated(!) Tests

Test "internal" TypeScript (JS) functionality

Test download (async) actions

Test performance:

- round trip times, latency, memory usage
- on different sized data sets 1, 5, 50, 500, 5000, ...
- of size 50KB, 500KB, 5MB,...
- optional (concurrency): multiple clients

https://mochajs.org https://jasmine.github.io

Deployment (CI, CD)

John Feiner

Staging

Compiler Optimisation: TS->JS

Optimise: Minify, Combine

Obfuscate Code

Staging (rollback):

- Development / Deployment servers

Toolchain - Summary

John Feiner

Toolchain - Summary

Git, coding, programming language, IDE, testing, deployment:

TypeScript -> JavaScript, ...

HTML5, CSS3, ...

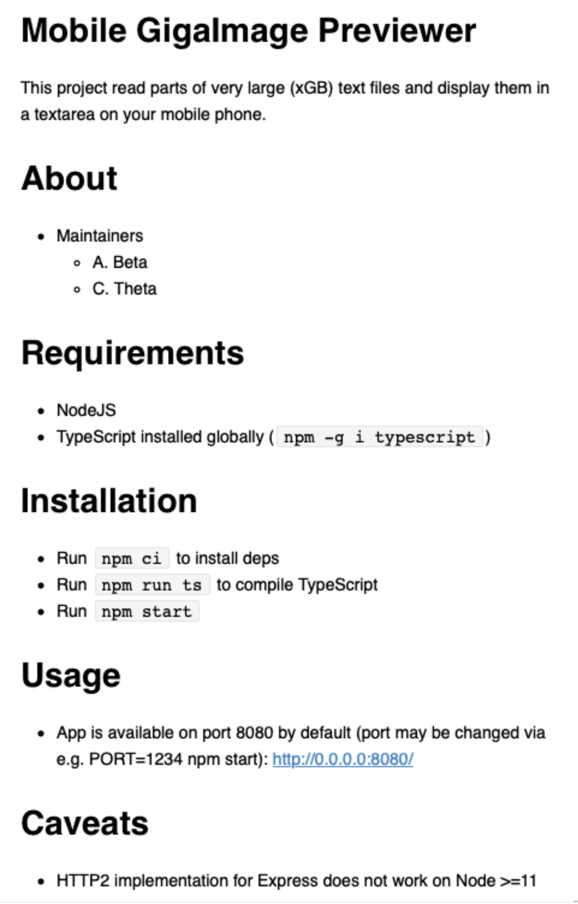
Testing performance and memory footprint:

Prepare test suite to measure downloads of 5, 50, 500 images.

Later, you might run static source code analysis:

Style guide checks, code metrics

Simple Demo "Readme.md"



Mobile GigaImage Previewer

This project read parts of very large (xGB) text files and display them in a textarea on your mobile phone.

About

- Maintainers
  - A. Beta
  - C. Theta

Requirements

- NodeJS
- TypeScript installed globally (`npm -g i typescript`)

Installation

- Run `npm ci` to install deps
- Run `npm run ts` to compile TypeScript
- Run `npm start`

Usage

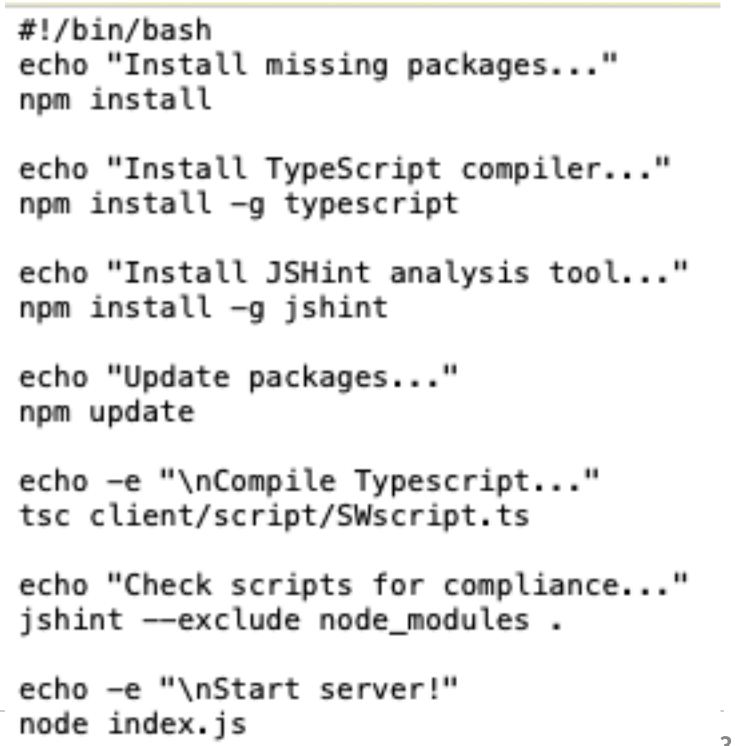
- App is available on port 8080 by default (port may be changed via e.g. PORT=1234 npm start): http://0.0.0.0:8080/

Caveats

- HTTP2 implementation for Express does not work on Node >=11

Simple Demo "startup.sh"

```
...
curl https://eoimages.....C1.png \
      > server/public/images/image.png
curl http://..../big.txt > server/public/big.txt
...
cd client && yarn install && yarn build &&
cd ..
cd server && yarn install && yarn start cd ..
```



```
#!/bin/bash
echo "Install missing packages..."
npm install

echo "Install TypeScript compiler..."
npm install -g typescript

echo "Install JSHint analysis tool..."
npm install -g jshint

echo "Update packages..."
npm update

echo -e "\nCompile Typescript..."
tsc client/script/SWscript.ts

echo "Check scripts for compliance..."
jshint --exclude node_modules .

echo -e "\nStart server!"
node index.js
```