

# **Useful technologies in software projects**

Adam Petříček

# Introduction

- not mandatory, but useful
- any software project
- enhancing best practices
- used to:
  - improve code quality
  - save time
  - eliminate mistakes
  - make collaborative projects easier

# Code documentation

- function docs comments
  - can be used for generating wiki
  - code is more understandable by other people
  - necessity for great open-source projects
- API endpoint documentation – **Swagger**
  - code formatted as YAML / JSON
  - generates nicely looking API structure
  - endpoints can be tested

```
/**
 * Sum the given two numbers and return the result
 * You can sum integers and floats
 *
 * @param int/float $num1
 * @param int/float $num2
 * @param mixed $third_param
 * @return int/float
 */
function sum($num1, $num2, $third_param)
{
    return $num1 + $num2;
}
```

```

1 swagger: "2.0"
2 info:
3   description: "Tool used for PDF generation from different templates"
4   version: "0.1.0"
5   title: "PDF Generator"
6 host: "pdf-generator.dt.jablotron.cz"
7 basePath: "/api/"
8 schemes:
9   - http
10 tags:
11   - name: "v1/"
12     description: ""
13 produces:
14   - application/pdf
15 paths:
16   /v1/generate-single-sync:
17     post:
18       tags:
19         - "v1/"
20       summary: "Generates one PDF file from provided HTML template"
21       description: ""
22       parameters:
23         - in: body
24           name: json
25           description: JSON of all properties
26           schema:
27             type: object
28             required:
29               - html
30             properties:
31               html:
32                 type: string
33                 description: "Full HTML code of template to generate PDF from"
34               header:
35                 type: string
36                 description: "Full HTML code of header to be inserted in front of HTML template"
37               footer:
38                 type: string
39                 description: "Full HTML code of header to be inserted behind HTML template"
40               fileName:
41                 type: string

```

# PDF Generator 0.1.0

[ Base URL: pdf-generator.dt.jablotron.cz/api/ ]

Tool used for PDF generation from different templates

Schemes

HTTP

## v1/

POST

/v1/generate-single-sync Generates one PDF file from provided HTML template

Parameters

Try it out

Name	Description
json	JSON of all properties
object	Example Value   Model
(body)	<pre>{   "html": "string",   "header": "string",   "footer": "string",   "fileName": "string",   "password": "string", }</pre>

# Static code analysis

- Prettier, Linters (language-specified)

## 1. **catching bugs** - Linters

- no unused, undeclared variables
- no class reassigning
- force return in class getter

## 2. **code formatting** - Prettier

- normalized number of spaces
- limit characters in one line
- keyword spacing

- code formatting useful in collaborative projects

```
1  if (condition1)
2  {
3      //...
4  }else if (condition2) {
5      //...
6  } else
7  {
8      //...
9  }
10
11
12
13  if (condition1) {
14      //...
15  }
16  else if (condition2) {
17      //...
18  }
19  else {
20      //...
21  }
```

# Code testing

- writing program that expects designated result for each case that can happen in your code
- example - **testing a function that moves robot by some distance forward**
  - 1. case: `distance = 10`
    - you expect robot to move by 10 units
  - 2. case: `distance = "hello"`
    - you expect program to throw exception
  - 3. case: `distance = 0`
    - you expect robot to stay in place
- if all cases match the expectations, tests succeed
- rewriting tests everytime you change your code

# Generated code coverage

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Line #
All files	76.92	69.23	68.42	80.65	
src	100	100	100	100	
index.js	100	100	100	100	
src/application	66.67	50	33.33	80	
App.js	66.67	50	33.33	80	58
src/application/About	50	100	0	50	
index.js	50	100	0	50	4
src/application/Users	0	100	0	0	
index.js	0	100	0	0	3-4
src/design-system/Alert	0	100	0	0	
index.js	0	100	0	0	3
src/design-system/GlobalStyles	100	50	100	100	
index.js	100	50	100	100	21
src/design-system/Message	100	100	100	100	
index.js	100	100	100	100	
src/design-system/Toggle	100	83.33	100	100	
index.js	100	83.33	100	100	12
src/utilities	83.33	100	83.33	85.71	
addOne.js	0	100	0	0	1

# Docker

- OS-level virtualization (OS not virtualized)
  - isolates software to containers - **containerization**
  - used for having the same working environment for software on different machines
- 
1. you create **dockerfile** (set of instructions) for the app where you specify all the needed files, libraries, etc.
  2. on different machine, you create **docker container** from the dockerfile where your app will be running
  3. you don't have to worry about mismatched versions of libraries, etc.



```
1 FROM dregistry.jablotron.cz/docker-images/node:lts-alpine
2
3 RUN mkdir -p /opt/node-app
4
5 WORKDIR /opt/node-app
6
7 RUN apk add --no-cache \
8     libstdc++=8.3.0-r0 \
9     qpdf=8.4.2-r0 \
10    libxrender \
11    libxext \
12    ca-certificates \
13    fontconfig \
14    ttf-dejavu \
15    ttf-droid \
16    && update-ms-fonts \
17    && fc-cache -f \
18    && rm -rf /tmp/* \
19    && apk del .build-deps
20
21 COPY . .
22 RUN npm install
23
24 CMD ["npm", "run", "dev"]
```

# Git CI

- Continuous Integration
- GitHub and GitLab each with their own implementation
- way to automate routine tasks like testing, builds or deployment with every **push**
- CI can be set to do different tasks depending on commit tag
- you can generate **artifacts** for download - results of CI (code coverage)
- previously mentioned technologies can be automated using CI

# Git CI - practice example

- *setup:*

1. creating tests, static code analysis, containerization
2. adding these tasks to the CI

- *when you push to Git:*

1. if static code analysis or tests fail, commit gets rejected
2. when everything passes, app build starts (Docker)
3. after app is built, you can deploy it to production server automatically

**Any questions?**

# Questions

1. Which of the mentioned technologies seems the most useful to you? Why?
2. What other practices / technologies do you use for improving your software projects?
3. Do you believe that well written code needs comments? Why?
4. How would you describe some of your biggest bad habits while writing code? What can you do to get rid of them?

**Thanks for you attention**