Spis treści

[**1.** **RWD** 4](#_Toc508370202)

[1.1. max-device-width ogarnąć 4](#_Toc508370203)

[**2.** **CSS** 5](#_Toc508370204)

[2.1. BEM 5](#_Toc508370205)

[2.2. Kaskadowość 5](#_Toc508370206)

[2.3. Dziedziczenie 5](#_Toc508370207)

[2.4. Specyficzność 5](#_Toc508370208)

[**3.** **Bootstrap** 5](#_Toc508370209)

[3.1. kiedy i dlaczego używać 5](#_Toc508370210)

[**4.** **SASS** 5](#_Toc508370211)

[4.1. po co używać 6](#_Toc508370212)

[4.2. loader do webpacka 6](#_Toc508370213)

[4.3. co ma ponad CSS 6](#_Toc508370214)

[4.4. zobaczyć LESS – co to i podstawowe różnice do SASS 6](#_Toc508370215)

[4.5. SCSS 6](#_Toc508370216)

[**5.** **GIT i jego funkcje** 6](#_Toc508370217)

[5.1. Mergowanie 6](#_Toc508370218)

[5.2. Branchowanie 6](#_Toc508370219)

[5.3. pull requesty 6](#_Toc508370220)

[5.4. cofanie się do commitów 6](#_Toc508370221)

[5.5. Stash 7](#_Toc508370222)

[5.6. Konflikty - skąd się biorą i jak je rozwiązywać 7](#_Toc508370223)

[5.7. poprosić Ozza 7](#_Toc508370224)

[5.8. git flow 7](#_Toc508370225)

[**6.** **Npm, a yarn** 7](#_Toc508370226)

[6.1. popytać Ozza 7](#_Toc508370227)

[**7.** **JS** 7](#_Toc508370228)

[7.1. Ify 7](#_Toc508370229)

[7.2. Pętle 7](#_Toc508370230)

[7.3. Funkcje 8](#_Toc508370231)

[7.4. Prototypy 8](#_Toc508370232)

[7.5. operatory logiczne 8](#_Toc508370233)

[7.6. undefined vs null 8](#_Toc508370234)

[7.7. Asynchroniczność 8](#_Toc508370235)

[**8.** **JQuery** 8](#_Toc508370236)

[8.1. Funkcjonalności 8](#_Toc508370237)

[8.2. dlaczego wypadł z łask 8](#_Toc508370238)

[8.3. zalety na swój czas 8](#_Toc508370239)

[**9.** **OOP** 8](#_Toc508370240)

[9.1. dlaczego obiektowość 8](#_Toc508370241)

[9.2. implementacja w react 8](#_Toc508370242)

[**10.** **Ajax** 8](#_Toc508370243)

[10.1. ajax a redux 8](#_Toc508370244)

[10.2. po co używać 8](#_Toc508370245)

[10.3. zady i walety 8](#_Toc508370246)

[**11.** **API** 8](#_Toc508370247)

[11.1. api na stronach 8](#_Toc508370248)

[11.2. api do mongo w mern 9](#_Toc508370249)

[11.3. hash użytkownika API 9](#_Toc508370250)

[**12.** **Node** 9](#_Toc508370251)

[12.1. przypomnieć sobie jak postawić serwer na czystym node 9](#_Toc508370252)

[12.2. różnice z php 9](#_Toc508370253)

[12.3. back-end – serwer i zapytania do niego 9](#_Toc508370254)

[**13.** **ReactJS** 9](#_Toc508370255)

[13.1. szablony (PuG EJS) 9](#_Toc508370256)

[13.2. struktura danych 9](#_Toc508370257)

[13.3. cykl życia komponentu 9](#_Toc508370258)

[13.4. ReactDOM 9](#_Toc508370259)

[13.5. state vs props 9](#_Toc508370260)

[13.6. dlaczego React 9](#_Toc508370261)

[13.7. podstawowe różnice z AgularJS 9](#_Toc508370262)

[**14.** **ES6** 9](#_Toc508370263)

[14.1. co wprowadza 9](#_Toc508370264)

[14.2. jaki sposób kompiluje się do es2015 (babel) 9](#_Toc508370265)

[**15.** **Webpack** 9](#_Toc508370266)

[15.1. Ozz 10](#_Toc508370267)

[15.2. Pluginy 10](#_Toc508370268)

[15.3. Loadery 10](#_Toc508370269)

[15.4. HMR 10](#_Toc508370270)

[**16.** **ExpressJS** 10](#_Toc508370271)

[16.1. Middleware 10](#_Toc508370272)

[16.2. routing – endpointy 10](#_Toc508370273)

[16.3. CRUD 10](#_Toc508370274)

[16.4. JSON w JS 10](#_Toc508370275)

[16.5. autoryzacja 10](#_Toc508370276)

[16.6. serwowanie plików 10](#_Toc508370277)

[**17.** **Inne** 10](#_Toc508370278)

[17.1. Makiety 10](#_Toc508370279)

[17.2. SCRUM 10](#_Toc508370280)

[17.3. Polling 10](#_Toc508370281)

[17.4. Streaming 10](#_Toc508370282)

[17.5. Long-polling 10](#_Toc508370283)

[17.6. Socket.io 10](#_Toc508370284)

[**18.** **Redux** 11](#_Toc508370285)

[18.1. Flux 11](#_Toc508370286)

[18.2. Cqrs 11](#_Toc508370287)

[18.3. event sourcing 11](#_Toc508370288)

[18.4. zasady korzystania 11](#_Toc508370289)

[18.5. programowanie funkcyjne 11](#_Toc508370290)

[18.6. store w mern 11](#_Toc508370291)

[18.7. React-redux 11](#_Toc508370292)

[18.8. middleware redux 11](#_Toc508370293)

[18.9. data flow 11](#_Toc508370294)

[**19.** **React-router** 11](#_Toc508370295)

[19.1. co to jest i jak działa 11](#_Toc508370296)

[**20.** **MongoDB** 11](#_Toc508370297)

[20.1. nierelacyjne bazy danych 11](#_Toc508370298)

[20.2. SQL 11](#_Toc508370299)

[20.3. Mongoose 11](#_Toc508370300)

[20.4. operacje na bazie Mongo 11](#_Toc508370301)

[**21.** **MERN** 11](#_Toc508370302)

[21.1. inne wersje np. MEAN 12](#_Toc508370303)

[21.2. kompletnie inne technologie PERL 12](#_Toc508370304)

[21.3. aplikacje izomorficzne 12](#_Toc508370305)

[21.4. server side rendering 12](#_Toc508370306)

[21.5. wirtualizacja (VBox, vagrant, docker) – Ozz 12](#_Toc508370307)

[21.6. generator kodu 12](#_Toc508370308)

[21.7. Redux-thunk 12](#_Toc508370309)

[21.8. Fetch 12](#_Toc508370310)

[21.9. przejrzeć dokumentację 12](#_Toc508370311)

[**22.** **Pozostałe** 12](#_Toc508370312)

[22.1. REST API 12](#_Toc508370313)

[22.2. normalizacja danych 12](#_Toc508370314)

[22.3. ReactDnD 12](#_Toc508370315)

[22.4. compose i connect 12](#_Toc508370316)

[22.5. obadać punkty połączenia frontu z back-endem 12](#_Toc508370317)

[**23.** **Zasady** 12](#_Toc508370318)

[23.1. DRY 12](#_Toc508370319)

[23.2. KISS 12](#_Toc508370320)

[23.3. poziomy abstrakcji 13](#_Toc508370321)

[23.4. SOLID Principles 13](#_Toc508370322)

[**24.** **Wzorce projektowe** 13](#_Toc508370323)

[**25.** **Migracje** 13](#_Toc508370324)

[**26.** **Zagadnienia serwerowe** 13](#_Toc508370325)

[26.1. deployowanie, CI/CD 13](#_Toc508370326)

[26.2. Cache’owanie (memcache, redis, cloudflare) 13](#_Toc508370327)

[26.3. Domeny (DNS, Vhosty) 13](#_Toc508370328)

1. **RWD**

<https://www.w3schools.com/css/css_rwd_intro.asp>

* 1. max-device-width ogarnąć

DEPRECATED!!! Nie używać!

Rzeczywista ilość pikseli wyświetlacza (są np. wyświetlacze, w których na jeden piksel w CSS składają się 2 piksele).

1. **CSS**

<https://www.w3schools.com/css/default.asp>

* 1. BEM

<http://getbem.com/>

Block - Standalone entity that is meaningful on its own. – naming: .block

Element - A part of a block that has no standalone meaning and is semantically tied to its block. – naming: .block\_\_elem

Modifier - A flag on a block or element. Use them to change appearance or behavior. – naming: .block--mod, .block\_\_elem--mod

* 1. Kaskadowość

<https://developer.mozilla.org/en-US/docs/Web/CSS/Cascade>

Najpierw autor, potem użytkownik, potem przeglądarka. CSS Animations przed nimi. Przed tym wszystkim te z flagą !important user-agent, user, author.

* 1. Dziedziczenie

Properties such as color, font-family are inherited, and border, margin are not. Check if property is inherited by default when used. Can use inherit, initial and unset to set inheritance.

* 1. Specyficzność

<https://the-awwwesomes.gitbooks.io/html-css-step-by-step/content/pl/css-basics/css-specificity/index.html>

Selektory: najpierw #identyfikator, potem .klasa/pseudoklasa, potem element/pseudoelement. Im dłuższy (bardziej dokładny selektor) tym wyżej w hierarchii.

1. **Bootstrap**

<https://getbootstrap.com/>

* 1. kiedy i dlaczego używać

Używać kiedy stylowanie nie ma znaczenia. Np. dla potrzeb własnych, na potrzeby testów, dla aplikacji na użytek własny, dla prostych single page, które prawie nie mają elementów. Przyspiesza proces stylowania, ale tworzy generycznie wyglądający layout.

1. **SASS**

<https://sass-lang.com/>

* 1. po co używać

Dostępne są frameworki takie jak Bourbon, Compass, itp. To co oferuje ponad CSS (patrz punkt 4.3).

* 1. loader do webpacka

Należy dodać task runner do webpacka aby wykonywało się preprocesorowanie SASS do CSS

<https://www.jonathan-petitcolas.com/2015/05/15/howto-setup-webpack-on-es6-react-application-with-sass.html>

<https://jonathanmh.com/webpack-sass-scss-compiling-separate-file/>

* 1. co ma ponad CSS

Oferuje: zmienne, wielostopniowe zagnieżdżanie, mixiny, dziedziczenie i importowanie, użycie operatorów +, - ,\* , %, itp.

* 1. zobaczyć LESS – co to i podstawowe różnice do SASS

<https://x-coding.pl/blog/developers/sass-vs-less/>

* 1. SCSS

Posiada składnię kompatybilną z CSS (parametry w klamrach i oddzielone średnikami) co ułatwia przerobienie CSS.

1. **GIT i jego funkcje**

<https://git-scm.com/>

* 1. Mergowanie

Checkout na branch, do którego chcemy mergować

Merge brancha który chcemy mergować

Np. git checkout master

git merge hotfix

* 1. Branchowanie

git branch branch\_name

git checkout branch\_name

lub git checkout –b branch\_name

* 1. pull requesty

Podajemy z jakiego brancha na jakim repozytorium chcemy zmergować do jakiego branch na jakim repozytorium. Po code review i wprowadzeniu poprawek używamy merge pull request.

* 1. cofanie się do commitów

git log –oneline // pokazuje historię commitów z ich ID można wywołać git log branch\_name dla jednego brancha

git checkout commitID // przenosi nas do commita o danym ID

git revert HEAD~3 // reverts changes in 4th last commit and creates new commit after reverting, -n does not create new commit after reverting, branch\_name~5..branch\_name~2 zamiast HEAD cofa commity od 3 do 5 (included) na branchu branch\_name

git reset --hard commitID // cofa nas do commita o commitID i usuwa wszystkie commity, które były po nim z historii. Wystąpi problem przy próbie pusha. W takim przypadku lepiej użyć revert

* 1. Stash

git stash // usuwa niezcommitowane zmiany z working tree i umieszcza w schowku

git stash pop // usuwa to co jest w schowku i wprowadza do working tree

git stash apply // pozostawia zmiany w schowku i wprowadza do working tree

git stash list // pokazuje wszystkie dostępne istniejące stashe

git stash pop stash@{nr\_stosu} // przywraca to co jest w danym stosie i usuwa ten stos

* 1. Konflikty - skąd się biorą i jak je rozwiązywać

Przy mergowaniu jeżeli linia kodu w pliku została zmieniona już przez kogoś innego znim zmergowaliśmy naszą zmianę, lub plik został usunięty.

Należy przejść do pliku z konfliktem, rozwiązać konflikt ręcznie, usunąć markupy gita i scommitować wykonane zmiany, które rozwiązały konflikt. Można już kontynuować mergowanie. Przy usuniętym pliku należy dodać lub usunąć plik z naszego repozytorium (w zależności od decyzji jak rozwiązać konflik) i scommitoać zmianę.

* 1. poprosić Ozza

**…**

* 1. git flow

[**https://datasift.github.io/gitflow/IntroducingGitFlow.html**](https://datasift.github.io/gitflow/IntroducingGitFlow.html)

1. **Npm, a yarn**

<https://www.npmjs.com/>

<https://yarnpkg.com/lang/en/>

<https://www.keycdn.com/blog/npm-vs-yarn/>

* 1. popytać Ozza

**…**

1. **JS**
   1. Ify
   2. Pętle
   3. Funkcje
   4. Prototypy
   5. operatory logiczne
   6. undefined vs null
   7. Asynchroniczność
2. **JQuery**
   1. Funkcjonalności
   2. dlaczego wypadł z łask
   3. zalety na swój czas
3. **OOP**
   1. dlaczego obiektowość
   2. implementacja w react
4. **Ajax**
   1. ajax a redux
   2. po co używać
   3. zady i walety
5. **API**
   1. api na stronach
   2. api do mongo w mern
   3. hash użytkownika API
6. **Node**
   1. przypomnieć sobie jak postawić serwer na czystym node
   2. różnice z php
   3. back-end – serwer i zapytania do niego
7. **ReactJS**
   1. szablony (PuG EJS)
   2. struktura danych
   3. cykl życia komponentu
   4. ReactDOM
   5. state vs props
   6. dlaczego React
   7. podstawowe różnice z AgularJS
8. **ES6**
   1. co wprowadza
   2. jaki sposób kompiluje się do es2015 (babel)
9. **Webpack**
   1. Ozz
   2. Pluginy
   3. Loadery
   4. HMR
10. **ExpressJS**
    1. Middleware
    2. routing – endpointy
    3. CRUD
    4. JSON w JS
    5. autoryzacja
    6. serwowanie plików
11. **Inne**
    1. Makiety
    2. SCRUM
    3. Polling
    4. Streaming
    5. Long-polling
    6. Socket.io
12. **Redux**
    1. Flux
    2. Cqrs
    3. event sourcing
    4. zasady korzystania
    5. programowanie funkcyjne
    6. store w mern
    7. React-redux
    8. middleware redux
    9. data flow
13. **React-router**
    1. co to jest i jak działa
14. **MongoDB**
    1. nierelacyjne bazy danych
    2. SQL
    3. Mongoose
    4. operacje na bazie Mongo
15. **MERN**
    1. inne wersje np. MEAN
    2. kompletnie inne technologie PERL
    3. aplikacje izomorficzne
    4. server side rendering
    5. wirtualizacja (VBox, vagrant, docker) – Ozz
    6. generator kodu
    7. Redux-thunk
    8. Fetch
    9. przejrzeć dokumentację
16. **Pozostałe**
    1. REST API
    2. normalizacja danych
    3. ReactDnD
    4. compose i connect
    5. obadać punkty połączenia frontu z back-endem
17. **Zasady**
    1. DRY
    2. KISS
    3. poziomy abstrakcji
    4. SOLID Principles
18. **Wzorce projektowe**
19. **Migracje**
20. **Zagadnienia serwerowe**
    1. deployowanie, CI/CD
    2. Cache’owanie (memcache, redis, cloudflare)
    3. Domeny (DNS, Vhosty)