Spis treści

[**1.** **RWD** 5](#_Toc508204830)

[1.1. max-device-width ogarnąć 5](#_Toc508204831)

[**2.** **CSS** 5](#_Toc508204832)

[2.1. BEM 5](#_Toc508204833)

[2.2. Kaskadowość 5](#_Toc508204834)

[2.3. Dziedziczenie 5](#_Toc508204835)

[2.4. Specyficzność 5](#_Toc508204836)

[**3.** **Bootstrap** 5](#_Toc508204837)

[3.1. kiedy i dlaczego używać 5](#_Toc508204838)

[**4.** **SASS** 6](#_Toc508204839)

[4.1. po co używać 6](#_Toc508204840)

[4.2. loader do webpacka 6](#_Toc508204841)

[4.3. co ma ponad CSS 6](#_Toc508204842)

[4.4. zobaczyć LESS – co to i podstawowe różnice do SASS 6](#_Toc508204843)

[4.5. SCSS 6](#_Toc508204844)

[**5.** **GIT i jego funkcje** 6](#_Toc508204845)

[5.1. Mergowanie 6](#_Toc508204846)

[5.2. Branchowanie 6](#_Toc508204847)

[5.3. pull requesty 6](#_Toc508204848)

[5.4. cofanie się do commitów 6](#_Toc508204849)

[5.5. Stash 6](#_Toc508204850)

[5.6. Konflikty - skąd się biorą i jak je rozwiązywać 6](#_Toc508204851)

[5.7. poprosić Ozza 6](#_Toc508204852)

[5.8. git flow 7](#_Toc508204853)

[**6.** **Npm, a yarn** 7](#_Toc508204854)

[6.1. popytać Ozza 7](#_Toc508204855)

[**7.** **JS** 7](#_Toc508204856)

[7.1. Ify 7](#_Toc508204857)

[7.2. Pętle 7](#_Toc508204858)

[7.3. Funkcje 7](#_Toc508204859)

[7.4. Prototypy 7](#_Toc508204860)

[7.5. operatory logiczne 7](#_Toc508204861)

[7.6. undefined vs null 7](#_Toc508204862)

[7.7. Asynchroniczność 7](#_Toc508204863)

[**8.** **JQuery** 7](#_Toc508204864)

[8.1. Funkcjonalności 7](#_Toc508204865)

[8.2. dlaczego wypadł z łask 7](#_Toc508204866)

[8.3. zalety na swój czas 7](#_Toc508204867)

[**9.** **OOP** 7](#_Toc508204868)

[9.1. dlaczego obiektowość 7](#_Toc508204869)

[9.2. implementacja w react 7](#_Toc508204870)

[**10.** **Ajax** 8](#_Toc508204871)

[10.1. ajax a redux 8](#_Toc508204872)

[10.2. po co używać 8](#_Toc508204873)

[10.3. zady i walety 8](#_Toc508204874)

[**11.** **API** 8](#_Toc508204875)

[11.1. api na stronach 8](#_Toc508204876)

[11.2. api do mongo w mern 8](#_Toc508204877)

[11.3. hash użytkownika API 8](#_Toc508204878)

[**12.** **Node** 8](#_Toc508204879)

[12.1. przypomnieć sobie jak postawić serwer na czystym node 8](#_Toc508204880)

[12.2. różnice z php 8](#_Toc508204881)

[12.3. back-end – serwer i zapytania do niego 8](#_Toc508204882)

[**13.** **ReactJS** 8](#_Toc508204883)

[13.1. szablony (PuG EJS) 8](#_Toc508204884)

[13.2. struktura danych 8](#_Toc508204885)

[13.3. cykl życia komponentu 8](#_Toc508204886)

[13.4. ReactDOM 8](#_Toc508204887)

[13.5. state vs props 8](#_Toc508204888)

[13.6. dlaczego React 9](#_Toc508204889)

[13.7. podstawowe różnice z AgularJS 9](#_Toc508204890)

[**14.** **ES6** 9](#_Toc508204891)

[14.1. co wprowadza 9](#_Toc508204892)

[14.2. jaki sposób kompiluje się do es2015 (babel) 9](#_Toc508204893)

[**15.** **Webpack** 9](#_Toc508204894)

[15.1. Ozz 9](#_Toc508204895)

[15.2. Pluginy 9](#_Toc508204896)

[15.3. Loadery 9](#_Toc508204897)

[15.4. HMR 9](#_Toc508204898)

[**16.** **ExpressJS** 9](#_Toc508204899)

[16.1. Middleware 9](#_Toc508204900)

[16.2. routing – endpointy 9](#_Toc508204901)

[16.3. CRUD 9](#_Toc508204902)

[16.4. JSON w JS 9](#_Toc508204903)

[16.5. autoryzacja 9](#_Toc508204904)

[16.6. serwowanie plików 9](#_Toc508204905)

[**17.** **Inne** 9](#_Toc508204906)

[17.1. Makiety 10](#_Toc508204907)

[17.2. SCRUM 10](#_Toc508204908)

[17.3. Polling 10](#_Toc508204909)

[17.4. Streaming 10](#_Toc508204910)

[17.5. Long-polling 10](#_Toc508204911)

[17.6. Socket.io 10](#_Toc508204912)

[**18.** **Redux** 10](#_Toc508204913)

[18.1. Flux 10](#_Toc508204914)

[18.2. Cqrs 10](#_Toc508204915)

[18.3. event sourcing 10](#_Toc508204916)

[18.4. zasady korzystania 10](#_Toc508204917)

[18.5. programowanie funkcyjne 10](#_Toc508204918)

[18.6. store w mern 10](#_Toc508204919)

[18.7. React-redux 10](#_Toc508204920)

[18.8. middleware redux 10](#_Toc508204921)

[18.9. data flow 10](#_Toc508204922)

[**19.** **React-router** 10](#_Toc508204923)

[19.1. co to jest i jak działa 10](#_Toc508204924)

[**20.** **MongoDB** 11](#_Toc508204925)

[20.1. nierelacyjne bazy danych 11](#_Toc508204926)

[20.2. SQL 11](#_Toc508204927)

[20.3. Mongoose 11](#_Toc508204928)

[20.4. operacje na bazie Mongo 11](#_Toc508204929)

[**21.** **MERN** 11](#_Toc508204930)

[21.1. inne wersje np. MEAN 11](#_Toc508204931)

[21.2. kompletnie inne technologie PERL 11](#_Toc508204932)

[21.3. aplikacje izomorficzne 11](#_Toc508204933)

[21.4. server side rendering 11](#_Toc508204934)

[21.5. wirtualizacja (VBox, vagrant, docker) – Ozz 11](#_Toc508204935)

[21.6. generator kodu 11](#_Toc508204936)

[21.7. Redux-thunk 11](#_Toc508204937)

[21.8. Fetch 11](#_Toc508204938)

[21.9. przejrzeć dokumentację 11](#_Toc508204939)

[**22.** **Pozostałe** 11](#_Toc508204940)

[22.1. REST API 11](#_Toc508204941)

[22.2. normalizacja danych 11](#_Toc508204942)

[22.3. ReactDnD 12](#_Toc508204943)

[22.4. compose i connect 12](#_Toc508204944)

[22.5. obadać punkty połączenia frontu z back-endem 12](#_Toc508204945)

[**23.** **Zasady** 12](#_Toc508204946)

[23.1. DRY 12](#_Toc508204947)

[23.2. KISS 12](#_Toc508204948)

[23.3. poziomy abstrakcji 12](#_Toc508204949)

[23.4. SOLID Principles 12](#_Toc508204950)

[**24.** **Wzorce projektowe** 12](#_Toc508204951)

[**25.** **Migracje** 12](#_Toc508204952)

[**26.** **Zagadnienia serwerowe** 12](#_Toc508204953)

[26.1. deployowanie, CI/CD 12](#_Toc508204954)

[26.2. Cache’owanie (memcache, redis, cloudflare) 12](#_Toc508204955)

[26.3. Domeny (DNS, Vhosty) 12](#_Toc508204956)

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**
   1. Mergowanie
   2. Branchowanie
   3. pull requesty
   4. cofanie się do commitów
   5. Stash
   6. Konflikty - skąd się biorą i jak je rozwiązywać
   7. poprosić Ozza
   8. git flow
2. **Npm, a yarn**
   1. popytać Ozza
3. **JS**
   1. Ify
   2. Pętle
   3. Funkcje
   4. Prototypy
   5. operatory logiczne
   6. undefined vs null
   7. Asynchroniczność
4. **JQuery**
   1. Funkcjonalności
   2. dlaczego wypadł z łask
   3. zalety na swój czas
5. **OOP**
   1. dlaczego obiektowość
   2. implementacja w react
6. **Ajax**
   1. ajax a redux
   2. po co używać
   3. zady i walety
7. **API**
   1. api na stronach
   2. api do mongo w mern
   3. hash użytkownika API
8. **Node**
   1. przypomnieć sobie jak postawić serwer na czystym node
   2. różnice z php
   3. back-end – serwer i zapytania do niego
9. **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
10. **ES6**
    1. co wprowadza
    2. jaki sposób kompiluje się do es2015 (babel)
11. **Webpack**
    1. Ozz
    2. Pluginy
    3. Loadery
    4. HMR
12. **ExpressJS**
    1. Middleware
    2. routing – endpointy
    3. CRUD
    4. JSON w JS
    5. autoryzacja
    6. serwowanie plików
13. **Inne**
    1. Makiety
    2. SCRUM
    3. Polling
    4. Streaming
    5. Long-polling
    6. Socket.io
14. **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
15. **React-router**
    1. co to jest i jak działa
16. **MongoDB**
    1. nierelacyjne bazy danych
    2. SQL
    3. Mongoose
    4. operacje na bazie Mongo
17. **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
18. **Pozostałe**
    1. REST API
    2. normalizacja danych
    3. ReactDnD
    4. compose i connect
    5. obadać punkty połączenia frontu z back-endem
19. **Zasady**
    1. DRY
    2. KISS
    3. poziomy abstrakcji
    4. SOLID Principles
20. **Wzorce projektowe**
21. **Migracje**
22. **Zagadnienia serwerowe**
    1. deployowanie, CI/CD
    2. Cache’owanie (memcache, redis, cloudflare)
    3. Domeny (DNS, Vhosty)