JavaScript – framework'i

Autor:

NodeJS + NPM

Autor:

Czym jest NodeJS?

Jest to środowisko pozwalające uruchamiać JavaScript poza przeglądarką.

Wykorzystuje silnik **JavaScript V8** - używany w przeglądarce: **Google Chrome**.

Technologia back-end'owa, pozwalająca **uruchamiać język JavaScript na serwerze**.

https://nodejs.org/en/download/

Czym jest NPM?

Główne idee NPM, czyli **Node Package Manager**:

- Łatwość publikacji i instalacji bibliotek.
- Proste zarządzanie zależnościami.
- Łatwe wersjonowanie aplikacji.
- Ustrukturyzowane zarządzenie projektem.

Pierwsze kroki z NPM

Tworzenie nowego projektu:

```
$ npm init
```

Dodawanie nowych skryptów:

```
"scripts": {
      "start": "node index.js",
      "test": "node test.js"
},
```

Uruchamianie skryptów:

```
$ npm run start
```

```
$ npm run test
```

ZADANIE 1

1. Zainicjuj nowy projekt za pomocą NPM

2. Przyjrzyj się strukturze pliku: package.json

Pierwsze kroki z NPM

Instalowanie zależności globalnie

```
$ npm install angular-cli -g
```

Instalowanie zależności lokalnie

```
$ npm install angular-cli
```

Instalowanie zależności lokalnie z dodaniem ich do package.json

```
$ npm install angular-cli --save
```

```
$ npm install webpack-dev-server --save-dev
```

Instalowanie zależności zawartych w package.json

```
$ npm install
```

ZADANIE 2

Wykorzystaj istniejący projekt z poprzedniego zadania

1. Dodaj bibliotekę jQuery

2. Biblioteka jQuery powinna zostać w przyszłości automatycznie zainstalowana po wybraniu polecenia

```
$ npm install
```

Webpack

Autor:

Czym jest Webpack?

Jest to najpopularniejsze narzędzie typu - **bundler**, używany w projektach JavaScript.

Potrafi on np. spakować wiele różnych typów zasobów do jednego wynikowego pliku.

Czym jest Single Page Application?

Single Page Application to technologia, która pozwala wyświetlać poszczególne elementy strony, bez potrzeby ponownego załadowania całej strony.

Autor:

Single Page Application vs Multi Page Application

Główne różnice:

- Sposób działania
- Wydajność
- Architektura
- SEO

https://bulldogjob.pl/news/354-single-page-application-kontra-klasyczne-podejscie

React + Create React App

Autor:

Virtual DOM

- Jest to wirtualna instancja całego drzewa DOM
- Przy zmianie porównywane są zmiany i wyszukiwane konkretne element

https://programmingwithmosh.com/react/react-virtual-dom-explained/

https://pl.reactjs.org/docs/faq-internals.html

Czym jest React?

- biblioteka do budowania interfejsów użytkownika
- pozwala w łatwy sposób zbudować aplikację SPA
- nie jest framework'iem JavaScript

Czym jest Create React App

Create React App: to narzędzie (stworzone również przez programistów Facebooka), które zapewnia szybki start w tworzeniu aplikacji React-owej. Oszczędza to czasochłonnej konfiguracji. Wystarczy uruchomić jedno polecenie, a aplikacja Create React App skonfiguruje wszystkie narzędzia potrzebne do rozpoczęcia projektu.

Tworzenie projektu:

```
$ npx create-react-app my-new-app
```

Uruchomienie aplikacji:

```
$ npm start
```

Komponenty klasowe i funkcyjne

- Komponenty klasowe tworzymy gdy potrzebujemy manipulować stanem komponentu lub jego cyklem życia
- Komponenty funkcyjne są to z reguły proste komponenty pozbawione skomplikowanej logi, których zadaniem jest wyświetlenie interfejsu

https://pl.reactjs.org/docs/components-and-props.html

JSX

- Jest to składnia wykorzystywana przez bibliotekę React.js która jest kompilowana do JavaScript’u za pomocą takich narzędzi jak Webpack
- Ułatwia tworzenie elementów DOM

https://pl.reactjs.org/docs/introducing-jsx.html

Zdarzenia React-a: synthethic events

https://pl.reactjs.org/docs/events.html

Autor:

Dostęp do elementów drzewa DOM

https://pl.reactjs.org/docs/refs-and-the-dom.html

Stan komponentu

- Stan komponentu (state) umozliwia kontrolowanie kiedy komponent ma się sam zaktualizować (ponownie wyrenderować)
- Za każdym razem gdy stan zostanie zaktualizowany za pomocą metody **this.setState()**, komponent zostanie zaktualizowany (updated), co ponownie uruchomi metodę **render(),** która wyświetli zaktuaalizowane dane
- W stanie (state) powinniśmy przechowywać dane TYLKO takie które wyświetlamy w **UI.** Do pozostałych operacji powinniśmy korzystać ze standardowych pól klasy np. **this.scrollPosition**

Komponenty kontrolowane oraz niekontrolowane

Komponent kontrolowany to taki którego właściwości (props-y) kontrolowane są przez stan jego rodzica.

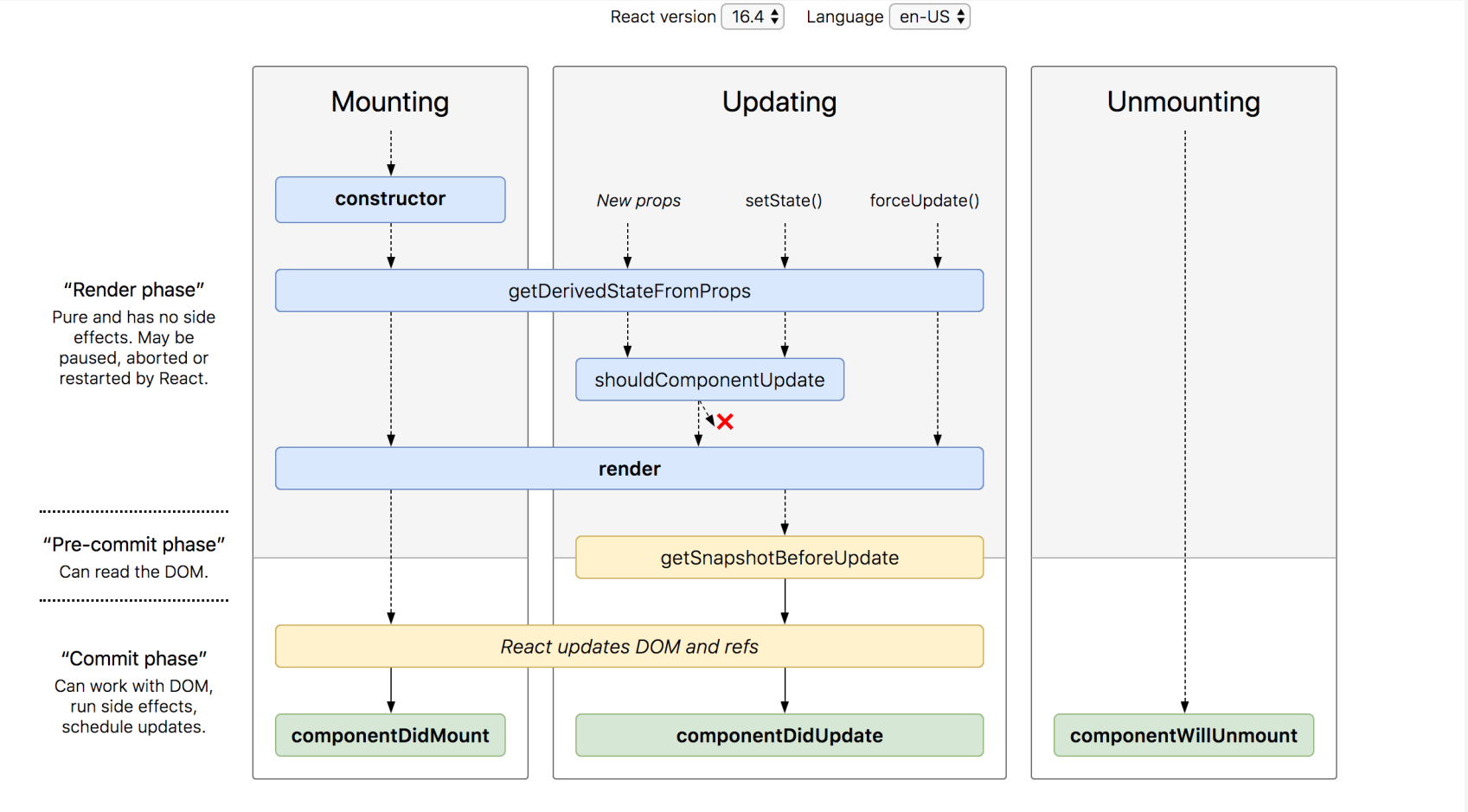
Komponent kontrolowany dzięki callback-owi orzymanego od rodzica modyfikuje jego stan (rodzica), tym samym otrzymując od rodzica nową wartość (props) do wyświetlenia.

https://medium.com/@AndrewBonner2/controlled-components-in-react-920f3e795d87

https://pl.reactjs.org/docs/uncontrolled-components.html

Autor:

Cyckle życia komponentu



https://pl.reactjs.org/docs/state-and-lifecycle.html

PureComponents oraz Memo

Używamy PureComponent oraz Memo w celach optymalizacyjnych, dzięki nim unikamy zbędnych przerenderowań - wasted renders.

https://pl.reactjs.org/docs/react-api.html#reactpurecomponent

https://pl.reactjs.org/docs/react-api.html#reactmemo

Autor:

Komponenty Wyższego Rzędu (HOC)

Jest to wzorzec służący do wydzielania logiki używanej przez wiele komponentów do jednego miejsca (DRY)

https://pl.reactjs.org/docs/higher-order-components.html

Kontekst API

Context API służy do przekazywania danych pomiędzy komponentami znajdującymi się w kompletnie różnych gałęziach drzewa

https://pl.reactjs.org/docs/context.html