JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 11 (2019) 145-148
Wysłane: 2018-12-02
Przyjęte: 2018-12-08

Porównanie narzędzi do tworzenia aplikacji typu SPA na przykładzie Ember i React

Jacek Wróbel*
Politechnika Lubelska, Instytut Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. W artykule przedstawiono wyniki porównania narzędzi do tworzenie aplikacji typu SPA przy użyciu platformy programistycznej Ember oraz React. Badania przeprowadzono z wykorzystaniem aplikacji testowych o takiej samej funkcjonalności, ale zaimplementowanych na obu platformach. Porównanie dotyczyło struktury projektu, wybranych metryk kodu oraz efektywności renderowania stron.

Słowa kluczowe: React; Ember; JavaScript

*Autor do korespondencji.

Adres e-mail: jacek.wrobel@pollub.edu.pl

Comparison of single-page application development using Ember and React example

Jacek Wróbel*
Institute of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract. The article presents the results of comparison of tools for creating SPA applications using the Ember and React framework. The research was carried out using test applications with the same functionality but implemented on both platforms. The comparison concerned the structure of the project, selected code metrics and the effectiveness of page rendering.

Keywords: React; Ember; JavaScript

*Corresponding author.

E-mail address: jacek.wrobel@pollub.edu.pl

1. Wstęp

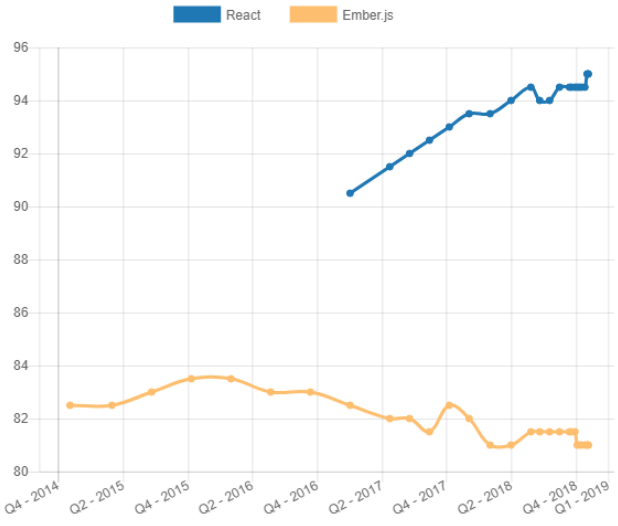
Język programowania JavaScript jest aktualnie bardzo szeroko wykorzystywany w dziedzinie informatyki, między innymi w tworzeniu stron internetowych, aplikacji internetowych, gier, aplikacji mobilnych itp. Można użyć go do zaprogramowania strony klienta, ale także strony serwera, np. za pomocą platformy programistycznej (ang. framework) Node.js [1]. Bardziej popularne jest jednak tworzenie widoku aplikacji, co było pierwotnym założeniem tego języka.

W międzyczasie wytworzono bardzo dużo nowych frameworków, które są mniej lub bardziej chętnie wykorzystywane przez programistów. Wykorzystywanie frameworków powoduje zwiększenie efektywności pracy programisty, poprawę jakości kodu oraz niezawodność. Tworzenie aplikacji na platformie programistycznej wymaga mniej linii kodu do napisania. Dodatkowo platformy takie są już odpowiednio zabezpieczone przed niechcianymi działaniami ze strony użytkowników. Wadą stosowania takich rozwiązań może być dość duża złożoność kodu oraz ich wydajność, w porównaniu do aplikacji napisanych w zwykłym JavaScript.

Dla mniej doświadczonego programisty, decyzja o tym, który framework wybrać nie jest prosta. W sieci można znaleźć kilka artykułów porównujących cechy wybranych frameworków JavaScript [9][10]. Zdania na temat, która platforma programistyczna jest lepsza są podzielone.

2. Cel badań

W niniejszym artykule zostaną przeanalizowane dwie platformy programistyczne: starsza Ember.js oraz nowsza biblioteka React,js. Są one obecnie jednymi z najpopularniejszych narzędzi wspierających budowę zaawansowanych aplikacji w JavaScript. Według portalu HotFrameworks - Ember jest na siódmym miejscu, zaś React na drugim miejscu w rankingach popularności (Rys. 1) [2].



Rys. 1. Porównanie popularności React i Ember w ostatnich 4 latach [2]

3. React i Ember

Ember wykorzystuje architekturę MVC oraz pozwala tworzyć aplikacje z wykorzystaniem:

- Ember CLI (ang. Command Line Interface) – konsola użytkownika wspomagająca tworzenie aplikacji;
- silnika szablonów Handlebars, który umożliwia wielokrotne wykorzystanie jednego komponentu;
- warstw danych, co daje możliwość spójnej komunikacji z zewnętrznymi API (ang. Application Programming Interface).

React jest biblioteką JavaScript, która posiada specyficzne dla frameworka cechy. Podobnie jak Ember, korzysta z komponentów, które są niezależnymi modułami aplikacji, jak np. formularz czy tabela danych. W każdym komponencie zdefiniowana jest funkcja *render(),* która wywoływana jest w razie zmiany stanu komponentu, aby go wyrenderować [5].

Zaletą React jest też język JSX, który jest pochodną XML [6] i pozwala ograniczyć liczbę wierszy kodu wpisywanych przez programistę. Przykład 1 i 2 pokazuje kod potrzebny do wyświetlenia hiperłącza w React i z wykorzystaniem języka JSX.

Przykład 1. Wyświetlenie hiperłącza za pomocą React

```
render: function() {
    return React.DOM.a({
        href: 'http://facebook.github.io/react'
    }, 'React');
}
```

Przykład 2. Wyświetlenie hiperłącza za pomocą JSX

```
render: function() {
return
    <a href="http://facebook.github.io/react">React</a> ;
}
```

W obu przykładach efektem końcowym będzie kod HTML: React

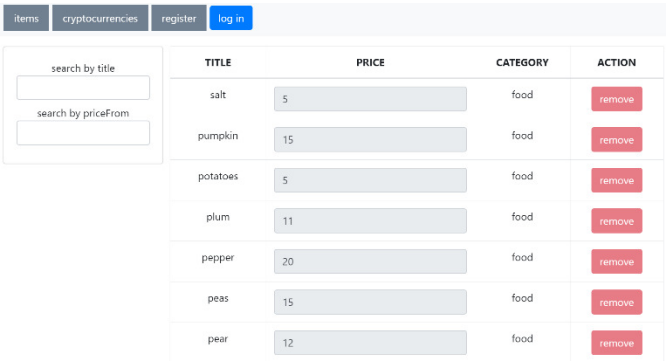
Najważniejsze cechy obu platform zostały zestawione w tabeli 1.

Tabela 1. Podstawowe właściwości React.js oraz Ember.js

	React	**Ember.js**
Autor	Jordan Walke	Yehuda Katz, Tom Dale, Charles Jolle
Język programowania	JavaScript	
Pierwsze wydanie	1 marca 2013	8 grudnia 2011
Aktualna wersja stabilna	16.3.2 (16 kwietnia 2018)	2.18.0 (1 stycznia 2018)
Licencja	MIT	

4. Aplikacje testowe

Aplikacje testowe zostały wykonane w środowisku programistycznym PhpStorm z zastosowaniem wtyczek do React oraz Ember [3]. Do zarządzania danymi zostało wykorzystane zewnętrzne API, dzięki któremu można dodać, usunąć lub edytować wpisy [4]. Aplikacje posiadają funkcjonalności CRUD (ang. Create, Read, Update, Delete), czyli realizują funkcje tworzenia, odczytu, edycji i usuwania danych. Interfejs graficzny użytkownika (Rys. 2) jest taki sam w obu przypadkach, dzięki czemu można łatwo porównać szybkość renderowania stron.



Rys. 2. Zrzut ekranu z aplikacji testowej

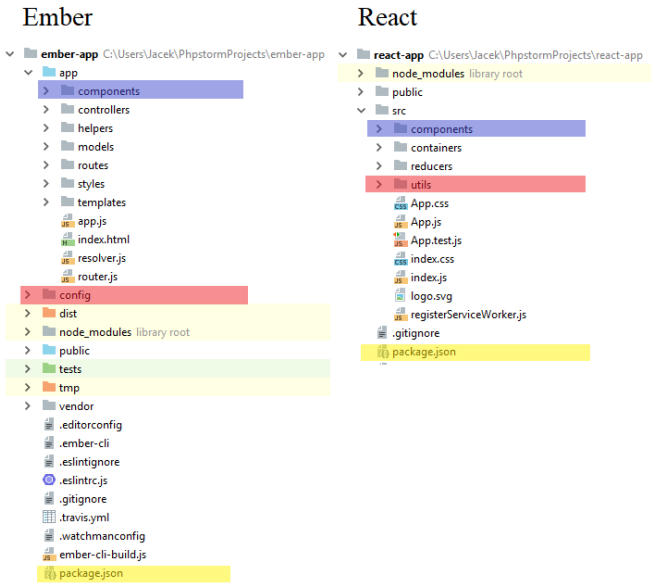
5. Analiza porównawcza

W analizie porównano:

- strukturę projektu;
- metryki kodu;
- szybkości renderowania stron.

5.1. Struktura aplikacji

W strukturze aplikacji Ember i React można zauważyć części wspólne, jak zaznaczono na rysunku 3. Na niebiesko oznaczone są komponenty aplikacji, m.in. tabela danych. Kolorem czerwonym zakreślono katalog z plikami konfiguracyjnymi, na żółto plik *package.json*, który zawiera wszystkie zależności potrzebne do zbudowania projektu.



Rys. 3. Porównanie struktury plików w obu aplikacjach testowych

5.2. Metryki kodu

Umieszczenie dużej liczby danych w tabeli wymaga zastosowania iteracji. W przypadku React wykorzystano funkcję *map()* (Przykład 3), natomiast w Ember użyta została pętla *{{#each}}*, do której przekazano dane (Przykład 4).

Przykład 3. Wypełnienie tabeli danymi w React

```
{this.props.data.map((item, idx) => {
                    return <tr key={idx}>
                       {this.props.config.map((field, idx) => {
                          return                                    <td
key={idx}>{item[field.key]}</td>
                       })}
                    </tr>
                 })}
```

Przykład 4. Wypełnienie tabeli danymi w Ember

```
{{#each cryptocurrencies as |cryptocurrency|}}
            <tr>
               <td>{{cryptocurrency.name}}</td>
               <td>{{cryptocurrency.symbol}}</td>
            </tr>
         {{/each}}
```

5.3. Szybkość renderowania

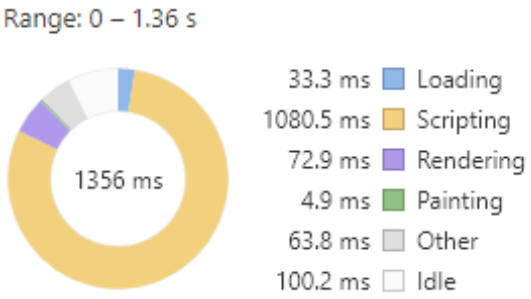
Testy wydajnościowe zostały przeprowadzone na dwóch platformach. Konfiguracja pierwszej platformy (laptop) jest następująca:

- procesor: Intel(R) Core(TM) i5-6200U CPU @ 2.30GHz, 2401 MHz, Rdzenie: 2, Procesory logiczne: 4,
- karta graficzna: Intel(R) HD Graphics 520,
- pamięć RAM: 8GB,
- system operacyjny: Windows 10 Home.

Specyfikacja sprzętowa drugiej platformy (PC) stanowi:

- procesor: Intel(R) Core(TM) i5-3570K CPU @ 3.40GHz, 3401 MHz, Rdzenie: 4, Procesory logiczne: 4,
- karta graficzna: AMD Radeon HD 7900 Series,
- pamięć RAM: 16GB,
- system operacyjny: Windows 10 Pro.

Wszystkie testy zostały przeprowadzone na przeglądarce Google Chrome, w której znajdują się narzędzia deweloperskie. Posiada ona wiele przydatnych funkcjonalności, między innymi możliwość uruchomienia testów wydajności. Wynik testu wydajności ładowania strony został przedstawiony na rysunku 4.



Rys. 4. Przykładowy wynik testu wydajności ładowannia strony

Testy wydajnościowe na małej liczbie danych

W tabelach 2 i 3 zostały przedstawione wyniki badania wydajności na około 180 rekordach w aplikacji React.js na laptopie i PC. Badane były czasy renderowania aplikacji oraz całkowity czas ładowania.

Tabela 2. Wyniki testów wydajnościowych na małej liczbie danych w React na pierwszej platformie

Numer próby	Czas rysowania aplikacji	Całkowity czas ładowania
1	127.8ms	1638ms
2	138.8ms	1954ms
3	144.2ms	1973ms
4	148.8ms	2136ms
5	137.4ms	1739ms
Średni czas	**139.4ms**	**1888ms**

Tabela 3. Wyniki testów wydajnościowych na małej liczbie danych w React na drugiej platformie

Numer próby	Czas rysowania aplikacji	Całkowity czas ładowania
1	95.2ms	1229ms
2	93.7ms	957ms
3	97.3ms	960ms
4	92.7ms	1130ms
5	95.6ms	1122ms
Średni czas	**94.9ms**	**1080ms**

Wyniki badań szybkości renderowania dla Ember.js ukazane są w tabelach 4 i 5.

Tabela 4. Wyniki testów wydajnościowych na małej liczbie danych w Ember na pierwszej platformie

Numer próby	Czas rysowania aplikacji	Całkowity czas ładowania
1	72.9ms	1356ms
2	87.0ms	1573ms
3	72.4ms	1467ms
4	79.8ms	1686ms
5	77.0ms	1631ms
Średni czas	**77.82ms**	**1543ms**

Tabela 5. Wyniki testów wydajnościowych na małej liczbie danych w Ember na drugiej platformie

Numer próby	Czas rysowania aplikacji	Całkowity czas ładowania
1	65.0ms	1080ms
2	74.4ms	1101ms
3	69.4ms	1029ms
4	70.5ms	997ms
5	70.6ms	1065ms
Średni czas	**69.98ms**	**1054ms**

Wyniki testów na małej liczbie danych wskazują na to, że Ember.js lepiej sobie radzi z rysowaniem aplikacji przy takiej liczbie rekordów (ok. 180 elementów), gdzie różnica średniego czasu na laptopie wynosiła około 63ms. Dodatkowo lepszy procesor dużo bardziej pomaga React.js, ponieważ średni czas na laptopie wynosił 139.4ms, zaś na PC już 94.9ms. W Ember.js ta różnica wynosiła około 8ms.

Testy wydajnościowe na dużych zbiorach danych

W tabelach 6 oraz 7 przedstawiono wyniki testów na dużych zbiorach danych, wynoszących ponad 2000 rekordów w aplikacji React na laptopie i PC.

Tabela 6. Wyniki testów wydajnościowych na dużej liczbie danych w React na pierwszej platformie

Numer próby	Czas rysowania aplikacji	Całkowity czas ładowania
1	219.5ms	2660ms
2	330.3ms	2976ms
3	226.0ms	2567ms
4	208.0ms	2818ms
5	228.8ms	2800ms
Średni czas	**242.52ms**	**2764ms**

Tabela 7. Wyniki testów wydajnościowych na dużej liczbie danych w React na drugiej platformie

Numer próby	Czas rysowania aplikacji	Całkowity czas ładowania
1	185.6ms	1870ms
2	199.1ms	1959ms
3	189.4ms	1908ms
4	187.1ms	1941ms
5	182.0ms	1857ms
Średni czas	**188.64ms**	**1907ms**

W tabelach 8 i 9 zostały zaprezentowane wyniki czasów rysowania dla aplikacji Ember kolejno na laptopie i PC.

Tabela 8. Wyniki testów wydajnościowych na dużej liczbie danych w Ember na pierwszej platformie

Numer próby	Czas rysowania aplikacji	Całkowity czas ładowania
1	595.1ms	4135ms
2	587.5ms	4004ms
3	908.7ms	4890ms
4	585.8ms	4392ms
5	499.7ms	4224ms
Średni czas	**635.36ms**	**4329ms**

Tabela 9. Wyniki testów wydajnościowych na dużej liczbie danych w Ember na drugiej platformie

Numer próby	Czas rysowania aplikacji	Całkowity czas ładowania
1	493.5ms	3789ms
2	482.6ms	3713ms
3	466.3ms	3656ms
4	460.3ms	3610ms
5	453.1ms	3657ms
Średni czas	**471.16ms**	**3685ms**

Wyniki na dużej liczbie danych, tj. 2000 rekordów, prezentują się nieco odmiennie od tych na małej liczbie danych, gdyż tym razem to React.js lepiej wypadł w czasie rysowania i ogólnym czasie ładowania aplikacji. Średni czas rysowania w React był ponad dwa razy mniejszy niż w Ember zarówno na laptopie jak i PC.

6. Wnioski

Po wykonaniu testów wydajnościowych można stwierdzić, że bardzo ważną rolę odgrywa sprzęt, na którym testowane są aplikacje, gdyż na lepszej konfiguracji sprzętowej czas renderowania w aplikacji był mniejszy o około 25%. Również ważną kwestią jest jaka liczba danych jest przetwarzana w aplikacji, ponieważ przy 180 rekordach w tabeli to Ember na obu platformach miał lepszy czas renderowania, zaś przy około 2000 rekordów React poradził sobie lepiej.

Dla aplikacji z małą liczbą danych to Ember.js wydaje się być lepszym frameworkiem. Jednak, gdy aplikacja musi wyświetlać dużą liczbę danych to React będzie lepszym wyborem. Zatem wybór technologii jest przede wszystkim zależny od tego, jaką aplikację programista chce stworzyć oraz tego, w jakim czasie będzie potrafił nauczyć się danego narzędzia.

Literatura

[1] A. Rauschmayer, Speaking JavaScript, O'Reilly, 2014.

[2] Wykorzystano wykres porównujący popularność frameworków, https://hotframeworks.com/languages/javascript/ [20.05.2018]

[3] https://www.jetbrains.com/phpstorm/ [20.05.2018]

[4] http://api.emitter.pl [20.05.2018]

[5] A. Banks, E, Porcello, Learning React, O'Reilly, 2017.

[6] https://reactjs.org/ [20.05.2018]

[7] https://www.emberjs.com/ [20.05.2018]

[8] https://coinmarketcap.com/api/ [28.11.2018]

[9] https://jcsi.pollub.pl/jcsi2016/volume2/jcsi2(2016)98-103.pdf [03.12.2018]

[10] https://blog.kollegorna.se/3-years-of-ember-6-months-of-react-34ce909a5ce1 [03.12.2018]