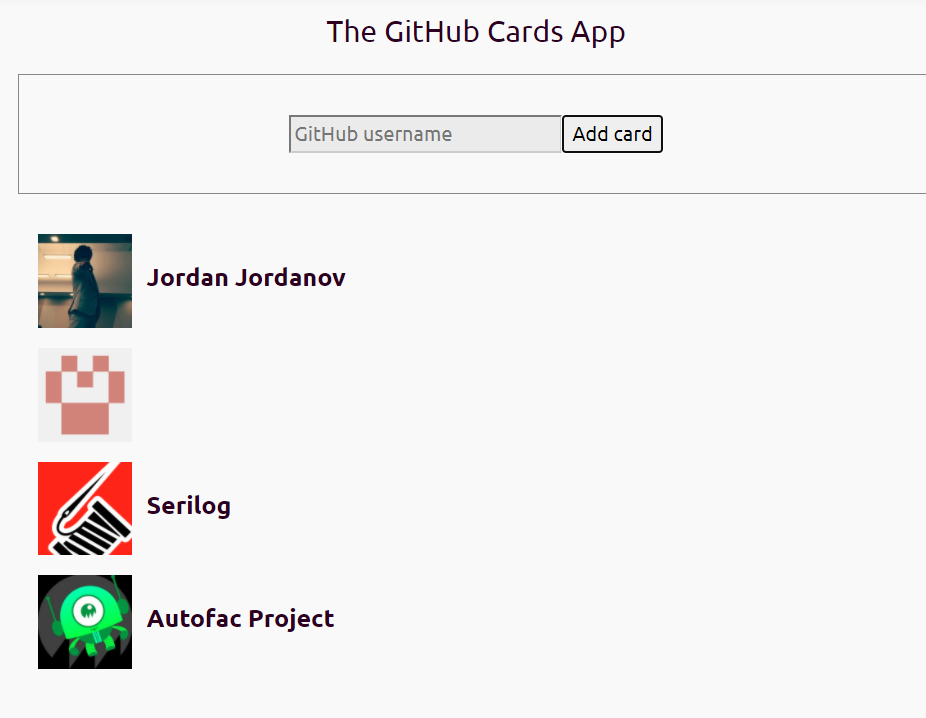
# The GitHub Cards App



***Какво ще изграждаме?***

Виждали сме прости компоненти, работили сме с множество компоненти и сме виждали как и кога да използваме състоянието и свойствата на даден компонент. Все още обаче не сме работили с реални данни. Ще направим точно това в този модул на курса и ще използваме публичния сървиз на GitHub. Ще изградим прост компонент на карта с профили на GitHub, който показва информация за списъци с вашите профили. Ще има формуляр, в който потребителят може да въведе име от GitHub и да използва „Добавяне на карта“ за да добави нов към списъка с показаните профили. Ще се учим как да приемаме входни данни от потребителя и как да ги използваме, за да осъществите повиквания към GitHub API. Целта на това приложение е да ви осигури комфортна работа с обекти от данни. Другото нещо за това приложение е, че ще използваме класови компоненти. А след това по-голямото приложение ще бъде изцяло написано с функционални компоненти и „куки“.

***React Class Components***

В <http://jsdrops.com/rgs2.1> подготвих малък компонент на функцията на приложението и го прикачих към DOM. Ще използваме този, за да съдържа всички останали компоненти в това приложение. Първото решение, което трябва да вземете в приложение на React е структурата на компонентите. Трябва да решите колко компонента да използвате и какво трябва да описва всеки. Това често е лесно, ако имате пълната картина на приложението, което създавате, но на практика нямате. Нашето приложение ще бъде списък с карти от GitHub. Първата ви улика, че имате нужда от компонент, който да представлява една карта, и друг компонент, който да представлява самия списък Нека започнем с преобразуване на този компонент в клас. Вместо функционален компонент, дефинирате клас със същото име и наследявате специален клас в React, React.Component. Наследяването на React.Component прави класа ви официален компонент на React. Ще научим две основни концепции в този пример: конструкто и ключовата дума this в класове. Всеки React компонент трябва да има функция за рендиране. Това е единствената необходима функция. Тя връща виртуалното DOM описание на вашия компонент. Създаваме истанции на класа и всяка получава своите свойства и състояние. Заглавието трябва да стане this.props.title. Създаваме класов компонент Card, който наследява React.Component, във функцията за рендиране връщаме div placeholder със клас github-профил.

Резултатът може да се провери на <https://jscomplete.com/playground/rgs2.2> .

***Работа с данни***

Под връзката <https://jscomplete.com/playground/rgs2.3> добавих временен масив testData, с който да работим. Тези данни бяха копирани от API на GitHub, така че след като приключим тестването на нашето приложение с тези фалшиви тестови данни, ще работи с реалните данни на API. URL адресът, който използвах за копиране на данните от GitHub, е api.github.com/users след това поставяте произволно потребителско име от GitHub и ще видите данните като JSON обект. Тъй като искаме да изобразим множество карти, се нуждаем от друг компонент, който да държи различните карти. Ще наречем този нов компонент CardList. Възможно е да управляваме състоянието на приложението в този компонент. Може да го създадем като функционален компонент. Функционалният компонент CardList ще получи стандартния аргумент за свойствата връща div, който ще съдържа нашия списък с карти. В този div ще изобразим компонент Card и ще трябва да променим компонента на приложението да изобрази CardList вместо Card. Нека поставим някои реални данни в компонента Card. За това ще използваме масива testData. Вътре в функцията за рендиране на компонента Card, ще създадем локална променлива. За изображението ще използваме profile.avatar\_url а за името се нуждаем от profile.name. А за името на компанията се нуждаем от profile.company. Компонента Card ще получава данните чрез свойствата. Един от начините да направите това е да вземете обекта, който държи свойствата и да го разнесете вътре в елемента на компонента Card.  
РЕЗУЛТАТ: <https://jscomplete.com/playground/rgs2.4> .

**Обобщение**

Изградихме няколко прости React компонента:

1) Card, за да визуализира информация за профил в GitHub,

2) CardList, за да преобразува масив от записи в масив от компоненти на Card

3) компонент форма за четене на данни от потребителя

4) компонент на приложение за управление на връзката между всички

# Създаване на локална среда за разработка

Създаването на среда за разработка не е сред най-забавните неща. Трябва да накарате много различни инструменти да работят заедно. Инструментите имат различни APIта и всеки инструмент ще трябва да бъде конфигуриран. Възможно е да срещнете проблеми като различни версии да не са съвместими и да не работят заедно. Цялата среда може внезапно да спре да работи след определено надграждане на инструментите и ще трябва да отделите известно време за отстраняване на грешки. Освен това средата за разработка се различава от продуктивната среда, което означава, че това, което работи за вас в разработката, може да не работи в производството. За щастие има някои инструменти на високо ниво, които можете да използвате, за да избегнете някои от кошмарите за работа със среди.

# Create React App

Проектът Create React App може да ви помогне да създадете локална среда за разработка на React с помощта на една команда. Не е нужно да научавате нови API или да поддържате нови конфигурации. За да имате локална среда за разработка, имате нужда от Node. Create React App е само npm пакет, а npm е Node Package Manager. Според документацията на Реакт ( <https://reactjs.org/docs/create-a-new-react-app.html> ) правилният начин за стартиране на ново приложение е:

**npx create-react-app stargame**

Инсталацията на пакетите ще отнеме около 2/3 минути.

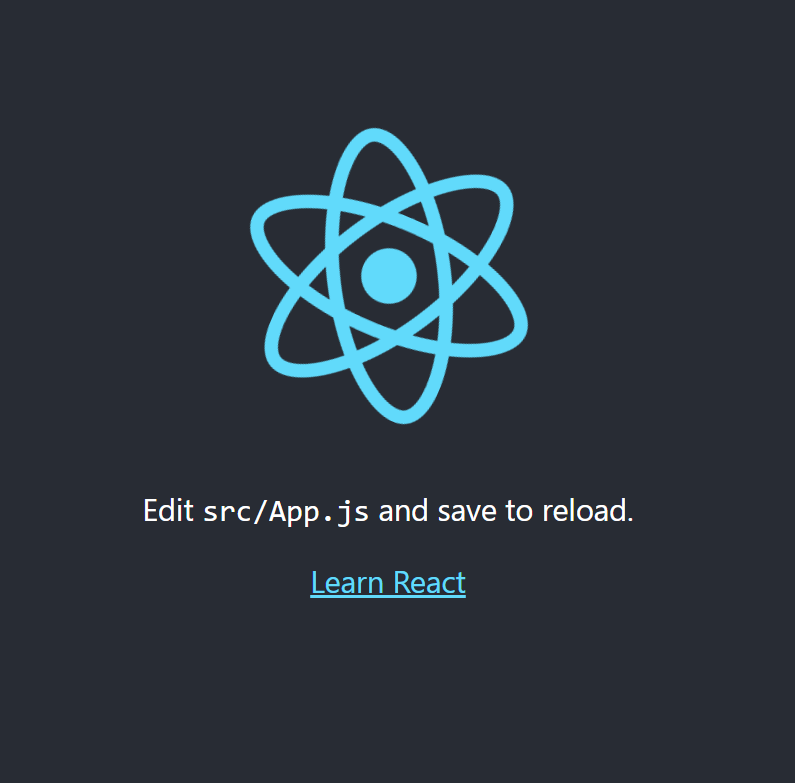
Влизаме в директорията на приложението:

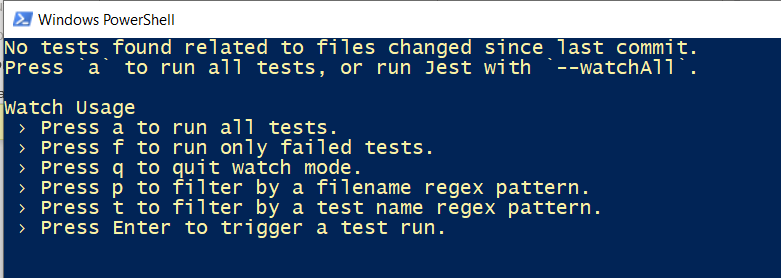
cd .\stargame\ или просто **cd stargame**.

Стартираме чисто новото ни приложение с:

**yarn start**

Очакваният резултат е този:

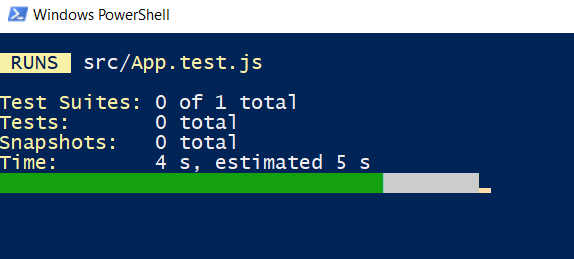


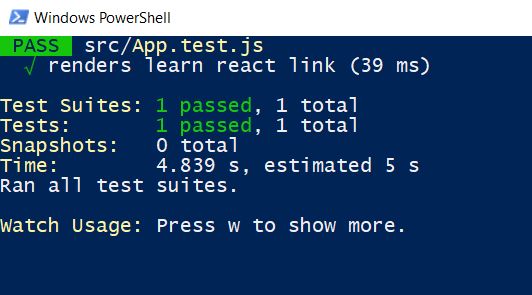
Прекъсваме процеса чрез командата от клавиши **Ctrl+C**, за да изпълним компонентните тестове:

**yarn test**

След това ще получиш следното съобщение:

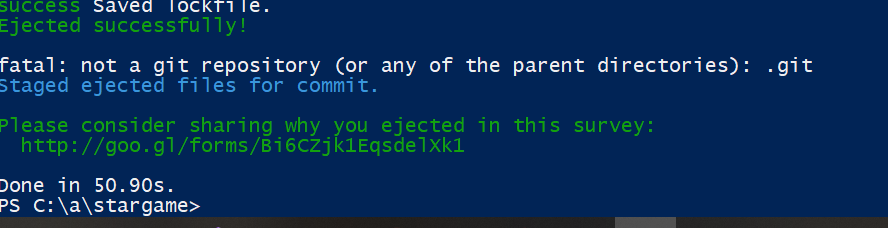
Въвеждаш **a** в конзолата.





Отново прекъсваме процеса. (Ctrl+C)

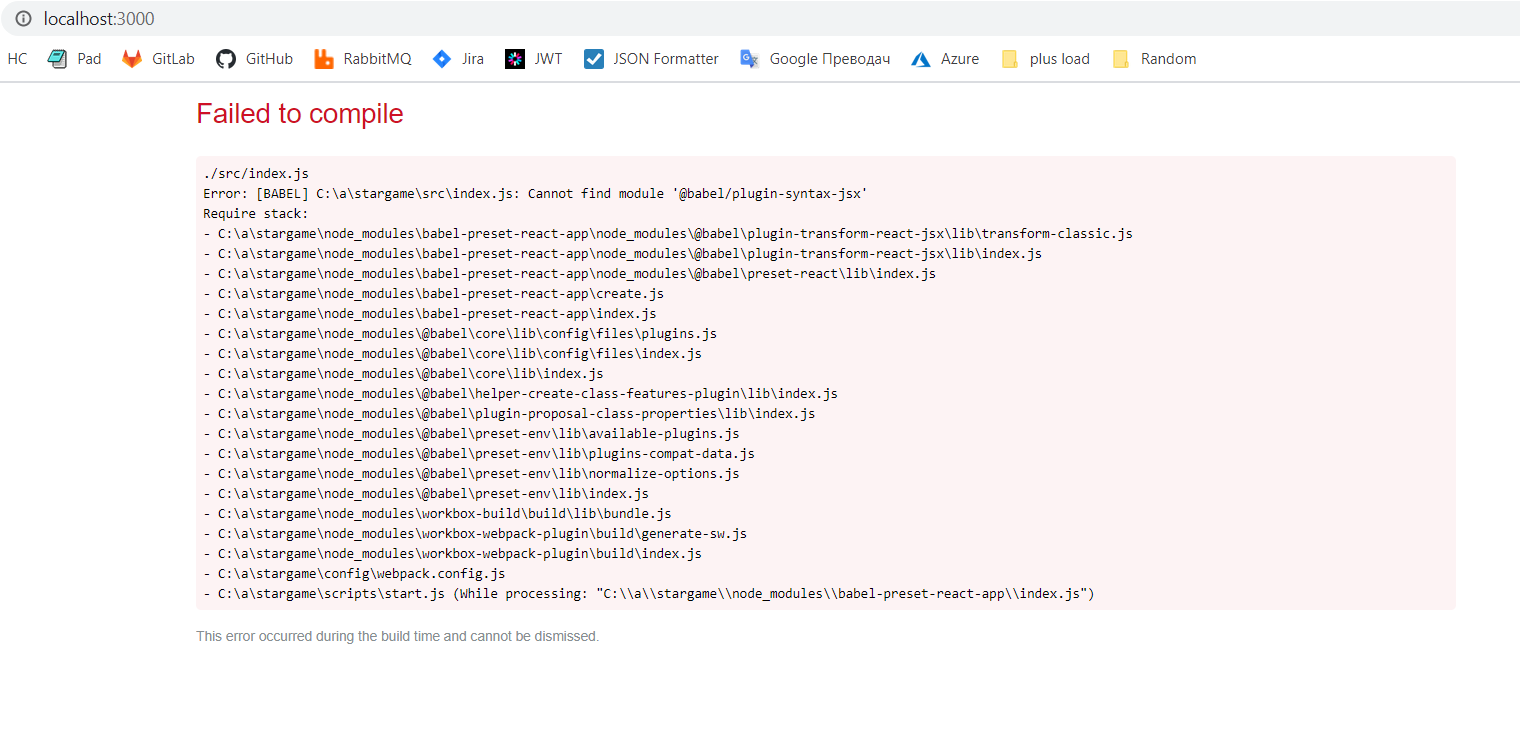
Изпълняваме командата :

**yarn eject**

С очакван резултат:

След това се уверете, че всичко е инсталирано правилно. (**yarn start**)

Възможно е, да се появи някой проблем като следния:



Грешката е добре описана. В случая е необходимо да изпълним:

**yarn add @babel/plugin-transform-react-jsx**

сададас