



Тема

Разработка на Preact и компарација со React. Анализа на брзина, техники за употреба, предности и негативности

Изработила:

Маја Јорданова 191125

Професор:

д-р Бобан Јоксимоски

Скопје, јуни 2023

Содржина

1. Вовед	3
2. Креирање и стартување на апликација во Preact наспроти React.....	3
3. Preact наспроти React.....	4
4. Пример на кратка апликација во React и Preact.....	6
5. Споредба на брзина	7
6. Предности на Preact.....	8
7. Предности на React	9
8. Недостатоци и ограничувања	10
9. Заклучок	12

1. Вовед

Светот на веб-развојот е трансформиран со појавата на модерни рамки и библиотеки на JavaScript, давајќи им можност на програмерите да создаваат софистицирани и интерактивни апликации.

Меѓу нив, React се појави како една од најпопуларните и нашироко усвоени рамки, давајќи им можност на девелоперите лесно да градат сложени интерфејси. Сепак, како што се зголемува побарувачката за побрзи и поедноставни решенија, се појави една поедноставна алтернатива наречена Preact.

Овој проект има за цел да го истражи развојот на Preact и да обезбеди сеопфатна споредба со React, фокусирајќи се на техниките за употреба, предностите и недостатоците. Со испитување на овие аспекти, програмерите ќе добијат вредни сознанија за компромисите помеѓу Preact и React, овозможувајќи им да донесуваат одлуки врз основа на нивните проектни барања.

2. Креирање и стартување на апликација во Preact наспроти React

Preact:

- Треба да имаме инсталирано Node.js
- Го отвараме терминалот и внесуваме:

```
npm install -g preact-cli
```

```
preact create default my-preact-app
```

```
cd my-preact-app
```

```
npm run dev
```

React:

```
npx create-react-app my-react-app
```

```
cd my-react-app
```

```
npm start
```

На овој начин стартуваме 2 апликации, една во Preact, друга во React.

3. Preact наспроти React

Preact е JavaScript библиотека дизајнирана за градење кориснички интерфејси. Има многу сличности со React, но со фокус на ефикасноста и брзината. Preact има за цел да обезбеди слично искуство за развој на React, притоа нудејќи помало влијание и побрзи перформанси.

Preact има мал виртуелен DOM и, со големина од само 3 KB, овозможува брз трансфер од сервер до клиент и е значително помала библиотека од React. Оваа компактна големина овозможува побрзо почетно време на вчитување, подобрени перформанси на уреди од ниска класа и подобри кориснички искуства, особено во области со побавна интернет конекција.

Preact нуди и компатибилност со екосистемот React. Ова значи дека програмерите запознаени со React можат лесно да преминат кон користење на Preact без да мора да научат сосема нов сет на концепти и API. Компатибилноста на

Considerations for Choosing Preact vs React

CONSIDERATION	PREACT	REACT
COMPLEXITY	Simpler API - may not offer all the advanced features of React	May be better for projects that require a more feature-rich and flexible API
SIZE	Smaller bundle size, can be beneficial for optimizing load times and reducing bandwidth usage	Larger bundle size
PERFORMANCE	Faster and more efficient virtual DOM implementation, making it good for high-performance apps	Virtual DOM implementation may be suitable for larger projects with more complex UI components

Preact им овозможува на програмерите да ги искористат постоечките компоненти, библиотеки и алатки на React, а со тоа Preact станува атрактивна опција за изработка на проекти.

И покрај помалата големина, Preact не ја ограничува функционалноста. Ги обезбедува повеќето основни карактеристики кои се наоѓаат во React, како што е виртуелен DOM за ефикасно прикажување, архитектура базирана на компоненти и модел на еднонасочен проток на податоци. Овие карактеристики им овозможуваат на програмерите да градат сложени кориснички интерфејси и да одржуваат чиста и организирана база на кодови.

Покрај тоа, Preact може да се пофали со едноставен и интуитивен API, што го прави лесен за учење и употреба. Неговиот рационализиран дизајн им овозможува на програмерите брзо да напредуваат и да започнат да градат апликации без непотребни сложености. Оваа едноставност не само што ја подобрува продуктивноста на програмерите, туку и го прави Preact одличен избор за мали до средни проекти или прототипови.

React од друга страна пак, е моќна и широко прифатена JavaScript библиотека за градење кориснички интерфејси. Успеа да направи револуција во развојот на предниот дел со воведување архитектура базирана на компоненти и виртуелен пристап за рендерирање DOM.

Една од клучните карактеристики што го издвојува React е неговиот виртуелен DOM. Наместо директно да манипулира со DOM-от на прелистувачот, React создава виртуелна

претстава на интерфејсот и ефикасно ги ажурира само неопходните делови кога ќе се случат промени. Овој пристап го намалува бројот на вистински DOM манипулации, што резултира со подобрени перформанси и понепречено корисничко искуство, особено во апликации со динамични и интерактивни елементи.

Preact - the better React.js?



VS.



Архитектурата базирана на компоненти на React е уште еден основен аспект на неговиот дизајн. Ги охрабрува програмерите да го разделат корисничкиот интерфејс на повеќекратни и самостојни компоненти, кои можат да се состават за да создадат сложени кориснички интерфејси. Оваа модуларност промовира повторна употреба, одржување и приспособливост на кодот, што го олеснува управувањето и ажурирањето на апликациите од големи размери.

Со одржување на јасна поделба помеѓу податоците и компонентите на корисничкиот интерфејс, React гарантира дека промените во состојбата на апликацијата се постапуваат на предвидлив начин. Ова помага во одржување на конзистентна состојба на апликацијата и го олеснува дебагирањето и тестирањето.

Понатаму, React има огромна и активна заедница, обезбедувајќи им на програмерите голема поддршка, ресурси и библиотеки од трети страни. Овој просперитетен екосистем нуди решенија за широк опсег на случаи на употреба, вклучувајќи рутирање, ракување со форми, управување со состојби и библиотеки со компоненти на UI. Тоа им овозможува на програмерите да ги користат веќе постоечките алатки и компоненти, забрзувајќи го развојот и промовирајќи го квалитетот на кодот.

Сепак, React не е без свои предизвици. Една од потенцијалните недостатоци на React е неговата поголема големина на пакетот во споредба со други поедноставни алтернативи. Сеопфатниот сет на функции и обемиот екосистем на React придонесуваат за неговата големина, што може да влијае на почетните времиња на вчитување, особено во сценарија со ограничена пропусност.

4. Пример на кратка апликација во React и Preact

```
// ReactApp.js
import React, { useState } from 'react';

const ReactApp = () => {
  const [count, setCount] = useState(0);

  const increment = () => {
    setCount(count + 1);
  };

  return (
    <div>
      <h1>React App</h1>
      <p>Count: {count}</p>
      <button onClick={increment}>Increment</button>
    </div>
  );
};

export default ReactApp;
```

```
// PreactApp.js
import { h, useState } from 'preact';

const PreactApp = () => {
  const [count, setCount] = useState(0);

  const increment = () => {
    setCount(count + 1);
  };

  return (
    <div>
      <h1>Preact App</h1>
      <p>Count: {count}</p>
      <button onClick={increment}>Increment</button>
    </div>
  );
};

export default PreactApp;
```

Во верзијата со **Preact**, можеме да забележиме неколку разлики:

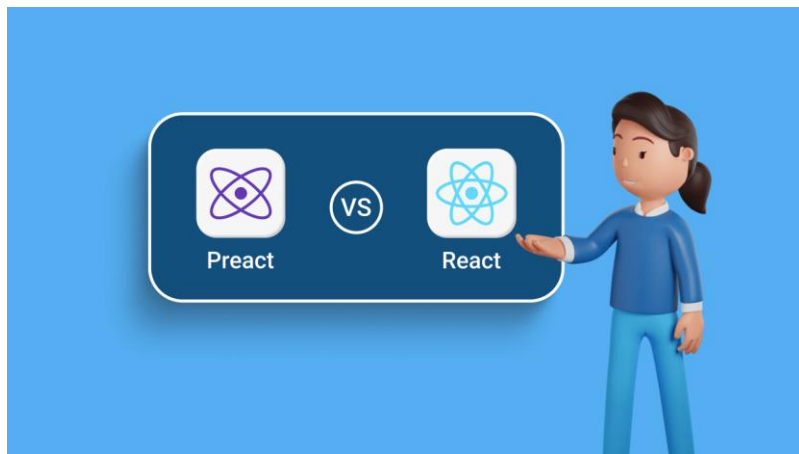
1. Наместо да вклучиме **'React'**, ние вклучуваме **'h'** од **'preact'**. **'h'** се користи како функција createElement во Preact.
2. **'useState'** се вклучува директно од **'preact'** наместо од **'react'**.
3. Целокупната структура и синтакса остануваат многу слични на верзијата React, со мали разлики во вклучувањето на некои од елементите.

Двете верзии ја постигнуваат истата функционалност – приказ на број и негово зголемување кога ќе се кликне на копчето **'Increment'**. Preact нуди помала големина на пакетот во споредба со React, при што честопати е повеќе посакувана алтернатива за проекти кои даваат приоритет на ограничувањата на перформансите и големината.

5. Споредба на брзина

Важно е да се забележи дека реалните перформанси на двете библиотеки може да се разликуваат врз основа на фактори како што се сложеноста на апликацијата, оптимизациите на прелистувачот и техниките за имплементација.

Preact, бидејќи е поедноставна библиотека, нуди забележителни предности во однос на



брзината. Помалата големина резултира со побрзо почетно време на вчитување, што го прави Preact одличен избор за проекти кои имаат приоритет за брзо прикажување и подобро корисничко искуство, особено во средини со низок опсег или ограничени ресурси. Намалените трошоци за виртуелната имплементација на DOM на

Preact придонесуваат за побрзо време на рендерирање, овозможувајќи брзи ажурирања на интерфејсот.

Од друга страна пак, природата богата со карактеристики и обемиот екосистем на React може да доведат до малку поголема големина на пакетот во споредба со Preact. Иако ова може да влијае на почетното време на вчитување, React компензира со својот ефикасен алгоритам за усогласување кој ги оптимизира ажурирањата на DOM. React интелигентно ги идентификува и ажурира само неопходните делови на корисничкиот интерфејс, што резултира со ефикасно прикажување и минимално влијание врз перформансите додека апликацијата се зголемува.

Важно е да се земе предвид дека разликата во перформансите помеѓу Preact и React може да не е значајна кај мали до средни апликации. Сепак, како што се зголемува сложеноста на апликацијата, оптимизациите на перформансите на React, како што е способноста за ефикасно справување со големите хиерархии на компоненти, може да станат поволни.

Алатките за бенчмаркинг може да се користат за мерење и споредување на перформансите на Preact и React во одредени сценарија. Овие алатки ги оценуваат метриците како што се времето на прикажување, користењето на меморијата и големината на пакетот. Резултатите може да варираат во зависност од конкретниот случај на употреба и оптимизациите што ги применуваат програмерите.

Сепак на крајот, изборот помеѓу Preact и React треба да ги земе предвид специфичните барања на проектот. Ако брзината и минималниот отпечаток се клучни фактори, едноставната природа на Preact и побрзото време на почетно вчитување може да бидат претпочитаниот избор. Меѓутоа, ако проектот вклучува сложени интерфејси со потреба од приспособливост и обемен екосистем, оптимизациите на перформансите на React и огромната поддршка од заедницата го прават силен конкурент.

Од споредбата помеѓу двете JavaScript библиотеки може да се заклучи дека помалата големина на Preact и побрзото почетно време на вчитување обезбедуваат предности во однос на брзината и ефикасноста на ресурсите.

6. Предности на Preact

Preact нуди неколку предности што го прават привлечен избор за програмери кои бараат едноставна и ефикасна JavaScript библиотека за градење кориснички интерфејси.

Помал отпечаток: Една од клучните предности на Preact е неговата помала големина во споредба со React. Минималниот отпечаток на Preact се претвора во побрзо време на вчитување, особено на уреди од ниска класа или во ситуации со побавни мрежни врски.

Компатибилност со React екосистемот: Preact е дизајниран да биде компатибилен со екосистемот React, што е значајна предност за програмерите кои веќе се запознаени со React. Оваа компатибилност им овозможува на програмерите беспрекорно да ги користат постоечките компоненти, библиотеки и алатки на React. Обезбедува лесна транзициона патека за мигрирање на проектите на React во Preact, овозможувајќи им на програмерите да имаат корист од брзината на Preact додека ја одржуваат компатибилноста со воспоставените решенија базирани на React.

Подобрени перформанси: Едноставната природа на Preact придонесува за неговите подобрени перформанси. Ефикасниот алгоритам за имплементација и усогласување на виртуелниот DOM на библиотеката овозможува брзо прикажување и минимално влијание врз перформансите како што се измерува апликацијата. Оваа предност е особено корисна за проекти кои даваат приоритет на одговорните и мазни кориснички интерфејси, обезбедувајќи брзо и беспрекорно корисничко искуство.

Продуктивност на програмерите: едноставноста и леснотијата на користење на Preact може да ја зголемат продуктивноста на програмерите. Библиотеката има јасен API и “плитка” крива на учење, што ја прави достапна за развивачите на различни нивоа на вештини. Неговиот минималистички пристап ја намалува сложеноста, дозволувајќи им на програмерите да се фокусираат на пишување чист и оддржлив код.

Активна заедница и поддршка: Иако Preact има помала заедница во споредба со React, сепак одржува активна и посветена корисничка база. Програмерите кои користат Preact можат да имаат корист од ресурсите управувани од заедницата, вклучувајќи документација, упатства и проекти со отворен код. Поддршката на заедницата поттикнува споделување знаење, решавање проблеми и соработка, овозможувајќи им на програмерите да ги надминат предизвиците и да учат еден од друг.

Флексибилност и приспособување: Preact нуди флексибилност и опции за прилагодување за да одговараат на различни проектни барања. Програмерите можат да ги изберат специфичните модули што им се потребни, овозможувајќи послаба и приспособена конструкција.

7. Предности на React

React нуди широк спектар на предности кои придонесоа за негово широко усвојување и популарност меѓу програмерите ширум светот. Еве неколку клучни предности од користењето на React за градење кориснички интерфејси:

Виртуелно прикажување DOM: виртуелната имплементација на DOM на React е камен-темелник на неговите перформанси и ефикасност. Со одржување на лесна виртуелна претстава на интерфејсот, React го оптимизира прикажувањето само со ажурирање на потребните компоненти кога ќе се појават промени. Овој пристап значително го намалува бројот на директни манипулации со вистинскиот прелистувач DOM, што резултира со побрзо и поефикасно ажурирање.

Архитектура базирана на компоненти: Архитектурата базирана на компоненти на React промовира повторна употреба на кодот, модуларност и одржливост. Разградувањето на корисничкиот интерфејс на компоненти за повеќекратна употреба поттикнува модуларен пристап кон развојот, каде што секоја компонента ја инкапсулира сопствената логика и рендерирање. Ова им овозможува на програмерите да градат сложени интерфејси со составување помали, самостојни компоненти, поедноставување на развојот, тестирањето и одржувањето.

Голем екосистем и заедница: React има придобивки од огромната и активна заедница на програмери, што доведе до богат екосистем од библиотеки, алатки и ресурси. Оваа просперитетна заедница обезбедува сеопфатна поддршка, обемна документација и бројни проекти со отворен код, овозможувајќи им на програмерите да ги искористат веќе постоечките решенија и да го забрзаат развојот.

Оптимизации за изведба: React вклучува оптимизација на перформансите, како што е неговиот алгоритам за усогласување, кој ефикасно го ажурира корисничкиот интерфејс со

идентификување и прикажување само на потребните промени. Овој пристап гарантира дека React може да се справи со големи и сложени хиерархии на компоненти без да ги “жртвува” перформансите. Дополнително, способноста на React да работи со рендерирање од страна на серверот (SSR) го олеснува подобреното време на почетно вчитување и придобивките од SEO (оптимизација на пребарувачот).

8. Недостатоци и ограничувања

Иако и Preact и React имаат свои силни страни, важно е да се земат предвид нивните недостатоци и ограничувања кога се донесува одлука за тоа која библиотека да се користи. Еве некои потенцијални недостатоци што треба да се имаат на ум:

Preact:

- **Намален сет на функции:** Preact, како едноставна алтернатива на React, може да има намален сет на функции во споредба со React. Одредени напредни функции или поретко користени API пронајдени во React можеби не се достапни во Preact. Ова ограничување може да влијае на проектите кои во голема мера се потпираат на специфичните функционалности на React.
- **Проблеми со компатибилноста:** Иако Preact има за цел да биде компатибилен со екосистемот React, може да има случаи кога одредени компоненти или библиотеки на React не се целосно компатибилни со Preact. Ова може да доведе до дополнителни напори за развој или заобиколување при интегрирање на постоечките бази на кодови React во проект на Preact.

React:



- **Крива на учење:** React има крива за учење, особено за програмери кои се нови во развојот на предниот дел или JavaScript. Разбирањето на концептите на React, како што се синтаксата на JSX и животниот циклус на компонентата, може да бара некои почетни напори. Оваа крива на учење може да биде предизвик за почетници, но станува помалку значајна бидејќи програмерите стекнуваат искуство и се запознаваат со React.
- **Поголема големина на пакетот:** сеопфатниот сет на функции и обемниот екосистем на React може да резултираат со поголема големина на пакетот во споредба со поедноставните алтернативи. Оваа поголема големина на пакетот може да влијае на почетното време на вчитување, особено во ситуации со ограничен опсег или побавни мрежни врски.

Општи размислувања:

- **Комплексност на проектот:** Изборот помеѓу Preact и React треба да ја земе предвид сложеноста на проектот. Иако поедноставната природа на Preact може да биде поволна за мали до средни проекти, робусноста и приспособливоста на React го прават посоодветен за сложени апликации кои бараат опширно управување со државата, напредно рутирање или ракување со големи хиерархии на компоненти.
- **Напори за миграција:** Мигрирањето на постоечки проект React во Preact или обратно може да внесе предизвици и дополнителни напори за развој. Разликите во API-ите, проблемите со компатибилноста и потенцијалното рефакторирање може да го направат процесот на миграција нетривијален и одзема многу време.

Неопходно е да се проценат овие недостатоци и ограничувања во контекст на специфичните барања и ограничувања пред започнување на самиот проект. Треба добро да се размисли за факторите како што се големината на проектот, потребите за изведба,

Comparing Key Features of Preact vs React

	 React	 Preact
SIZE	100KB	3KB
BUILD TIME	Slower	Faster
VIRTUAL DOM	✓	✓
ECOSYSTEM	Large	Small
JSX SUPPORT	✓	✓
PERFORMANCE	Good	Excellent
SERVER-SIDE RENDERING	✓	✓
COMMUNITY & RESOURCES	Large	Small

компатибилноста со постоечките бази на кодови и компромисите поврзани со секоја библиотека. Темелно истражување и прототипирање со Preact и React може да помогне при донесувањето на клучната одлука која најдобро се усогласува со целите и ограничувањата на проектот.

9. Заклучок

Preact и React нудат посебни предности и размислувања за програмерите кои градат кориснички интерфејси. Preact се издвојува како “лесна” алтернатива, обезбедувајќи побрзо време на вчитување, компатибилност со екосистемот React и ефикасно прикажување. Добро е прилагоден за проекти кои даваат приоритет на брзината, ефикасноста на ресурсите и беспрекорната интеграција со постоечките бази на кодови на React.

Од друга страна, React се истакнува во виртуелното DOM рендерирање, архитектура базирана на компоненти, обемни екосистеми и оптимизации за перформанси. Претставува популарен избор за градење сложени и скалабилни апликации.

Кога треба да се направи клучниот избор, односно да се одлучи помеѓу Preact и React, од клучно значење е да се проценат специфичните потреби на проектот кој треба да се изработи. Треба добро да се размисли за различни фактори како што се големината на проектот, барањата за изведба, компатибилноста со постоечките бази на кодови и кривата на учење поврзана со секоја библиотека. Ако брзината и помалиот отпечаток се најважни, Preact може да биде претпочитаниот избор. Меѓутоа, ако приспособливоста, сеопфатниот екосистем и оптимизациите на перформансите се клучни, React веројатно ќе биде посоодветна опција.

Притоа, треба да бидеме добро запознаени дека и двете библиотеки имаат свои ограничувања и компромиси. Preact може да има намален сет на функции во споредба со React, додека поголемата големина на пакетот и кривата на учење на React може да претставуваат предизвици. Дополнително, мигрирањето помеѓу двете библиотеки може да бара дополнителни напори за развој.

Но, на крајот изборот помеѓу Preact и React треба да се заснова на темелна евалуација на специфичните барања на проектот, мерејќи ги предностите и недостатоците на секоја библиотека. Без оглед на изборот, и Preact и React нудат моќни алатки за градење модерни, одговорни и интерактивни кориснички интерфејси, охрабрувајќи ги програмерите да создаваат исклучителни веб-искуства.