**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**KHOA CÔNG NGHỆ PHẦN MỀM**

Logo

Description automatically generated

**BÁO CÁO ĐỒ ÁN 1**

**ĐỀ TÀI:**

**Tìm Hiểu ReactJs – ExpressJs Framework NodeJs**

**SE121.M11**

Giáo viên hướng dẫn: **ThS. Nguyễn Công Hoan**

Sinh viên thực hiện: **Phạm Đăng Quang – 16521001**

Tp. Hồ Chí Minh, tháng 12/2021

**NHẬN XÉT (của giảng viên hướng dẫn)**

……………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………….……………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………….……………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………….……………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………….……………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………….……………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………….……………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………….……………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………….……………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………….……………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………….……………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………….

# **LỜI CẢM ƠN**

Lời đầu tiên em xin chân thành gửi lời cảm ơn đến trường Đại học Công Nghệ Thông Tin đã tạo mọi điều kiện cho em có môi trường học tập thật tốt, xin cảm ơn tất cả thầy đã hết lòng quan tâm, dạy dỗ và trang bị cho chúng em những kiến thức quý báu, cần thiết cho việc làm đồ án này cũng như công việc trong tương lai.

Em xin chân thành cảm ơn thầy Nguyễn Công Hoan đã tận tình hướng dẫn, chỉ bảo và giúp đỡ chúng em trong quá trình tìm hiểu, triển khai và thực hiện đề tài này. Trong thời gian học tập với thầy, em không ngừng tiếp thu thêm nhiều kiến thức bổ ích mà còn học tập được tinh thần làm việc, thái độ nghiên cứu nghiêm túc, hiệu quả, đây là những điều rất cần thiết cho em trong quá trình học tập và làm việc sau này.

Em cũng chân thành cảm ơn những thầy cô trường Đại học Công Nghệ Thông Tin nói chung và các thầy cô trong Khoa Công Nghệ Phần Mềm nói riêng đã trang bị cho em những kiến thức quý báu trong thời gian vừa qua.

Mặc dù em đã cố gắng hoàn thành đồ án này trong phạm vi khả năng cho phép nhưng vẫn không thể tránh khỏi những sai sót. Em mong cô và các bạn cảm thông và tận tình chỉ bảo.

Tp. Hồ Chí Minh, ngày 20 tháng 12 năm 2021

Contents

[**LỜI CẢM ƠN** 3](#_Toc91193257)

[**1. ReactJs** 5](#_Toc91193258)

[**1.1. Giới thiệu** 5](#_Toc91193259)

[**1.2. Tại sao nên chọn React** 5](#_Toc91193260)

[**1.3. Lược sử ra đời** 6](#_Toc91193261)

[**1.4. Các tính năng của React** 14](#_Toc91193262)

[**1.4.1. JSX – Javascript Syntax Extension** 14](#_Toc91193263)

[**1.4.2. Virtual DOM** 14](#_Toc91193264)

[**1.4.3. Performance** 20](#_Toc91193265)

[**1.4.4. Extensions** 20](#_Toc91193266)

[**1.4.5. One-way data binding** 21](#_Toc91193267)

[**1.4.6. Component Based Architecture** 21](#_Toc91193268)

[**1.5. So sánh giữa React với Vue.Js và Angular** 21](#_Toc91193269)

[**1.5.1. Angular** 21](#_Toc91193270)

[**1.5.2. Vue.js** 24](#_Toc91193271)

[**1.5.3. EmberJS** 26](#_Toc91193272)

[**1.6. Quy mô cộng đồng phát triển:** 27](#_Toc91193273)

[**1.7. Quy mô cộng đồng sử dụng:** 33](#_Toc91193274)

[**1.8. Kiến trúc** 34](#_Toc91193275)

[**2. ExpressJS** 37](#_Toc91193276)

[**2.1. ExpressJs là gì** 37](#_Toc91193277)

[**2.2. ExpressJs được dùng để làm gì** 37](#_Toc91193278)

[**2.3. Lược sử ra đời** 37](#_Toc91193279)

[**2.4. Lợi ích khi sử dụng ExpressJS** 44](#_Toc91193280)

[**2.5. Danh sách tính năng cơ bản** 44](#_Toc91193281)

[**2.6. Một số công nghệ tương đương** 45](#_Toc91193282)

[**2.6.1 Hapi.js** 45](#_Toc91193283)

[**2.6.2. Sails.js** 45](#_Toc91193284)

[**2.6.3. Koa.js** 46](#_Toc91193285)

[**2.7. Cộng đồng phát triển và sử dụng** 47](#_Toc91193286)

# **1. ReactJs**

## **1.1. Giới thiệu**

React là một trong những thư viện Javascript front-end phổ biến nhất trong lĩnh vực phát triển web. Nó được sử dụng bởi các công ty lớn, được thành lập lâu năm cho đến các công ty khởi nghiệp (Netflix, Airbnb, Instagram và The New York Times,…). React đem lại rất nhiều ưu điểm trong việc phát triển ứng dụng web cũng như mobile, khiến nó trở thành lựa chọn tốt hơn những framework khác như Angular.js

React là một thư viện Javascript được tạo ra nhằm xây dựng nhanh và giao diện người dùng (UI) dành cho ứng dụng web và mobile. React là một thư viện mã nguồn mở(open-source), nền tảng các thành phần(component-based), thư viện front-end chỉ chịu trách nhiệm cho view layer của ứng dụng.

Trong kiến trúc MVC (Model View Controller), view layer chịu trách nhiệm cho việc ứng dụng được nhìn và cảm nhận như thế nào. React được tạo bởi Jordan Walke, một kỹ sư phần mềm làm việc tại Meta.

## **1.2. Tại sao nên chọn React**

Sự phổ biến của React ngày nay đã làm lu mờ tất cả các framework phát phiển front-end khác. Dưới đây là lý do:

* **Dễ dàng tạo ra các dynamic applications**: React làm cho điều đó dễ dàng hơn trong việc tạo ra các ứng dụng web động bởi vì nó yêu cầu ít hơn trong việc codings và cung cấp nhiều chức năng hơn, trái ngược với Javascript, ngôn ngữ mà việc coding sẽ trở nên phức tạp rất nhanh.
* **Cải thiện hiệu suất**: React sử dụng Virtual DOM, do đó tạo các ứng dụng web nhanh hơn. Virtual DOM so sánh trạng thái trước đó của component và chỉ cập nhập lại các phần trong React DOM bị thay đổi, thay vì cập nhập lại tất cả các component một lần nữa, như các ứng dụng web thông thường hay làm.
* **Tái sử dụng component**: Component là nền tảng của bất kì ứng dụng React nào, và một ứng dụng bao gồm nhiều các component được hợp thành. Các component có các thành phần logic và xử lý riêng, và chúng có thể được sử dụng trong khắp ứng dụng, do đó giảm đáng kể thời gian phát triển của ứng dụng.
* **Luồng dữ liệu một chiều**: React sử dụng luồng dữ liệu một chiều. Điều này có nghĩa khi xây dựng ứng dụng React, lập trình viên thường lồng component con trong component cha. Khi luồng dữ liệu là một chiều, nó sẽ dễ dàng hơn trong việc debug lỗi và biết được vấn đề xảy ra ở đâu trong ứng dụng tại thời điểm được đề cập.
* **Quá trình học tập nhanh chóng**: React dễ học, vì nó hầu hết kết hợp giữa HTML cơ bản và các khái niệm Javascript với các lợi ích bổ sung. Tuy nhiên, trong trường hợp sử dụng với các công cụ và framework khác, chúng ta cần dành nhiều thời gian hơn để hiểu rõ về các thư viện React.
* **Có thể được sử dụng để phát triển ứng dụng web và mobile**: chúng ta thường biết tới React trong việc phát triển web nhưng đó không phải là tất cả mà React có thể làm được. React Native là một framework bắt nguồn từ React, rất phổ biến và được sử dụng để phát triển các ứng dụng các ứng dụng mobile đẹp mắt. Vì thế, trong thực tế, React có thể được sử dụng để phát triển cả ứng web và mobile.
* **Công cụ chuyên dụng để debug**: Meta đã phát hành một tiện ích mở rộng (extension) của chrome cho phép debug các ứng dụng React. Điều này làm cho tiến trình debug ứng dụng web và mobile React nhanh hơn và dễ dàng hơn.

## **1.3. Lược sử ra đời**

React được tạo bởi Jordan Walke, một kỹ sư phần mềm làm việc tại Meta, người đã phát hành một nguyên mẫu đầu tiên của React được gọi là “FaxJS”. Ông chịu ảnh hưởng của XHP, một thư viện thành phần HTML cho PHP. Nó được triển khai lần đầu tiên cho New’s Feed của Facebook năm 2011 và sau đó cho Instagram năm 2012. Nó được công bố mã nguồn tại JSConf US tháng năm 2013.

React Native cho phép phát triển trên Android, IOS và Universal Windows Platform (UMP) được công bố tại Facebook React Conf vào tháng 2 năm 2015 và được công bố mã nguồn tháng 3 năm 2015.

Vào 18 tháng 4 năm 2017, Facebook công bố React Fiber, một thuật toán nội bộ cho việc rendering, trái ngược với thuật toán render cũ của React, Stack. React Fiber trở thành nền tảng của bất kì cải tiến nào trong tương lai và phát triển các tính năng của thư việc React. Cú pháp thật sự để lập trình với React không thay đổi, chỉ có cách cú pháp được thực thi bị thay đổi. Hệ thống rendering cũ của React, Stack, được phát triển tại thời điểm mà sự theo dõi của hệ thống trên sự thay đổi động của ứng dụng không được xem trọng. Stack chậm chạp trong việc vẽ các animation phức tạp. Fiber chia các animation thành nhiều phân đoạn có thể được trải ra trên nhiều khung hình. Tương tự như vậy, cấu trúc của một trang có thể được chia thành nhiều phân đoạn có thể được bảo trì và cập nhập liên tục. Các hàm Javascript và đối tượng Virtual DOM được gọi là “fiber”, và mỗi cái có thể được vận hành và cập nhập riêng biệt, cho phép render trên màn hình mượt mà hơn.

Phiên bản:

|  |  |  |
| --- | --- | --- |
| **Version** | **Release Date** | **Changes** |
| 0.3.0 | 29 May 2013 | Initial Public Release |
| 0.4.0 | 20 July 2013 | Support for comment nodes <div>{/\* \*/}</div>, Improved server-side rendering APIs, Removed React.autoBind, Support for the key prop, Improvements to forms, Fixed bugs. |
| 0.5.0 | 20 October 2013 | Improve Memory usage, Support for Selection and Composition events, Support for getInitialState and getDefaultProps in mixins, Added React.version and React.isValidClass, Improved compatibility for Windows. |
| 0.8.0 | 20 December 2013 | Added support for rows & cols, defer & async, loop for <audio> & <video>, autoCorrect attributes. Added onContextMenu events, Upgraded jstransform and esprima-fb tools, Upgraded browserify. |
| 0.9.0 | 20 February 2014 | Added support for crossOrigin, download and hrefLang, mediaGroup and muted, sandbox, seamless, and srcDoc, scope attributes, Added any, arrayOf, component, oneOfType, renderable, shape to React.PropTypes, Added support for onMouseOver and onMouseOut event, Added support for onLoad and onError on <img> elements. |
| 0.10.0 | 21 March 2014 | Added support for srcSet and textAnchor attributes, add update function for immutable data, Ensure all void elements don't insert a closing tag. |
| 0.11.0 | 17 July 2014 | Improved SVG support, Normalized e.view event, Update $apply command, Added support for namespaces, Added new transformWithDetails API, includes pre-built packages under dist/, MyComponent() now returns a descriptor, not an instance. |
| 0.12.0 | 21 November 2014 | Added new features Spread operator ({...}) introduced to deprecate this.transferPropsTo, Added support for acceptCharset, classID, manifest HTML attributes, React.addons.batchedUpdates added to API, @jsx React.DOM no longer required, Fixed issues with CSS Transitions. |
| 0.13.0 | 10 March 2015 | Deprecated patterns that warned in 0.12 no longer work, ref resolution order has changed, Removed properties this.\_pendingState and this.\_rootNodeID, Support ES6 classes, Added API React.findDOMNode(component), Support for iterators and immutable-js sequences, Added new features React.addons.createFragment, deprecated React.addons.classSet. |
| 0.14.1 | 29 October 2015 | Added support for srcLang, default, kind attributes, and color attribute, Ensured legacy .props access on DOM nodes, Fixed scryRenderedDOMComponentsWithClass, Added react-dom.js. |
| 15.0.0 | 7 April 2016 | Initial render now uses document.createElement instead of generating HTML, No more extra <span>s, Improved SVG support, ReactPerf.getLastMeasurements() is opaque, New deprecations introduced with a warning, Fixed multiple small memory leaks, React DOM now supports the cite and profile HTML attributes and cssFloat, gridRow and gridColumn CSS properties. |
| 15.1.0 | 20 May 2016 | Fix a batching bug, Ensure use of the latest object-assign, Fix regression, Remove use of merge utility, Renamed some modules. |
| 15.2.0 | 1 July 2016 | Include component stack information, Stop validating props at mount time, Add React.PropTypes.symbol, Add onLoad handling to <link> and onError handling to <source> element, Add isRunning() API, Fix performance regression. |
| 15.3.0 | 30 July 2016 | Add React.PureComponent, Fix issue with nested server rendering, Add xmlns, xmlnsXlink to support SVG attributes and referrerPolicy to HTML attributes, updates React Perf Add-on, Fixed issue with ref. |
| 15.3.1 | 19 August 2016 | Improve performance of development builds, Cleanup internal hooks, Upgrade fbjs, Improve startup time of React, Fix memory leak in server rendering, fix React Test Renderer, Change trackedTouchCount invariant into a console.error. |
| 15.4.0 | 16 November 2016 | React package and browser build no longer includes React DOM, Improved development performance, Fixed occasional test failures, update batchedUpdates API, React Perf, and ReactTestRenderer.create(). |
| 15.4.1 | 23 November 2016 | Restructure variable assignment, Fixed event handling, Fixed compatibility of browser build with AMD environments. |
| 15.4.2 | 6 January 2017 | Fixed build issues, Added missing package dependencies, Improved error messages. |
| 15.5.0 | 7 April 2017 | Added react-dom/test-utils, Removed peerDependencies, Fixed issue with Closure Compiler, Added a deprecation warning for React.createClass and React.PropTypes, Fixed Chrome bug. |
| 15.5.4 | 11 April 2017 | Fix compatibility with Enzyme by exposing batchedUpdates on shallow renderer, Update version of prop-types, Fix react-addons-create-fragment package to include loose-envify transform. |
| 15.6.0 | 13 June 2017 | Add support for CSS variables in style attribute and Grid style properties, Fix AMD support for addons depending on react, Remove unnecessary dependency, Add a deprecation warning for React.createClass and React.DOM factory helpers. |
| 16.0.0 | 26 September 2017 | Improved error handling with introduction of "error boundaries", React DOM allows passing non-standard attributes, Minor changes to setState behavior, remove react-with-addons.js build, Add React.createClass as create-react-class, React.PropTypes as prop-types, React.DOM as react-dom-factories, changes to the behavior of scheduling and lifecycle methods. |
| 16.1.0 | 9 November 2017 | Discontinuing Bower Releases, Fix an accidental extra global variable in the UMD builds, Fix onMouseEnter and onMouseLeave firing, Fix <textarea> placeholder, Remove unused code, Add a missing package.json dependency, Add support for React DevTools. |
| 16.3.0 | 29 March 2018 | Add a new officially supported context API, Add new packagePrevent an infinite loop when attempting to render portals with SSR, Fix an issue with this.state, Fix an IE/Edge issue. |
| 16.3.1 | 3 April 2018 | Prefix private API, Fix performance regression and error handling bugs in development mode, Add peer dependency, Fix a false positive warning in IE11 when using Fragment. |
| 16.3.2 | 16 April 2018 | Fix an IE crash, Fix labels in User Timing measurements, Add a UMD build, Improve performance of unstable\_observedBits API with nesting. |
| 16.4.0 | 24 May 2018 | Add support for Pointer Events specification, Add the ability to specify propTypes, Fix reading context, Fix the getDerivedStateFromProps() support, Fix a testInstance.parent crash, Add React.unstable\_Profiler component for measuring performance, Change internal event names. |
| 16.5.0 | 5 September 2018 | Add support for React DevTools Profiler, Handle errors in more edge cases gracefully, Add react-dom/profiling, Add onAuxClick event for browsers, Add movementX and movementY fields to mouse events, Add tangentialPressure and twist fields to pointer event. |
| 16.6.0 | 23 October 2018 | Add support for contextType, Support priority levels, continuations, and wrapped callbacks, Improve the fallback mechanism, Fix gray overlay on iOS Safari, Add React.lazy() for code splitting components. |
| 16.7.0 | 20 December 2018 | Fix performance of React.lazy for lazily-loaded components, Clear fields on unmount to avoid memory leaks, Fix bug with SSR, Fix a performance regression. |
| 16.8.0 | 6 February 2019 | Add Hooks, Add ReactTestRenderer.act() and ReactTestUtils.act() for batching updates, Support synchronous thenables passed to React.lazy(), Improve useReducer Hook lazy initialization API. |
| 16.8.6 | 27 March 2019 | Fix an incorrect bailout in useReducer(), Fix iframe warnings in Safari DevTools, Warn if contextType is set to Context.Consumer instead of Context, Warn if contextType is set to invalid values. |
| 16.9.0 | 9 August 2019 | Add React.Profiler API for gathering performance measurements programmatically. Remove unstable\_ConcurrentMode in favor of unstable\_createRoot |
| 16.10.0 | 27 September 2019 | Fix edge case where a hook update wasn't being memoized. Fix heuristic for determining when to hydrate, so we don't incorrectly hydrate during an update. Clear additional fiber fields during unmount to save memory. Fix bug with required text fields in Firefox. Prefer Object.is instead of inline polyfill, when available. Fix bug when mixing Suspense and error handling. |
| 16.10.1 | 28 September 2019 | Fix regression in Next.js apps by allowing Suspense mismatch during hydration to silently proceed |
| 16.10.2 | 3 October 2019 | Fix regression in react-native-web by restoring order of arguments in event plugin extractors |
| 16.11.0 | 22 October 2019 | Fix mouseenter handlers from firing twice inside nested React containers. Remove unstable\_createRoot and unstable\_createSyncRoot experimental APIs. (These are available in the Experimental channel as createRoot and createSyncRoot.) |
| 16.12.0 | 14 November 2019 | React DOM - Fix passive effects (useEffect) not being fired in a multi-root app.  React Is - Fix lazy and memo types considered elements instead of components |
| 16.13.0 | 26 February 2020 | Features added in React Concurrent mode.  Fix regressions in React core library and React Dom. |
| 16.13.1 | 19 March 2020 | Fix bug in legacy mode Suspense.  Revert warning for cross-component updates that happen inside class render lifecycles |
| 16.14.0 | 14 October 2020 | Add support for the new JSX transform. |
| 17.0.0 | 20 October 2020 | "No New Features" enables gradual React updates from older versions.  Add new JSX Transform, Changes to Event Delegation |
| 17.0.1 | 22 October 2020 | React DOM - Fixes a crash in IE11 |
| 17.0.2 | 22 March 2021 | React DOM - Remove an unused dependency to address the SharedArrayBuffer cross-origin isolation warning. |

## **1.4. Các tính năng của React**

### **1.4.1. JSX – Javascript Syntax Extension**

JSX là một tiện ích mở rộng cho Javascript. Nó được sử dụng với React để diễn tả giao diện người dùng trông như thế nào. Bằng các sử dụng JSX, chúng ta có thể viết các cấu trúc HTML trong cùng một file chứa các code Javascript. Điều này làm cho code dễ hiểu và debug hơn, vì nó tránh việc sử dụng các cấu trúc Javascript DOM phức tạp.

jsx

### **1.4.2. Virtual DOM**

Khi tiếp cận với React chúng ta bắt gặp khái niệm Virtual DOM nhưng trước tiên cần làm rõ DOM là gì? **DOM** là tên gọi tắt của **Doccument Object Model** (Mô hình đối tượng tài liệu) là một chuẩn mực được xây dựng bởi W3C dùng để truy xuất và thao tác trên code HTML hay XML bằng các ngôn ngữ lập trình thông dịch (scripting language) như Javascript.

**DOM** giúp thao tác với dữ liệu theo mô hình hướng đối tượng do các phần tử trong DOM có cấu trúc được định nghĩa thành các đối tượng, phương thức, thuộc tính để có thể truy xuất dễ dàng. Chúng được coi như các node và được biểu diễn dưới dạng DOM tree.

Diagram

Description automatically generated

Nguồn: W3C School

Trong khi HTML là một đoạn code, DOM là một thể hiện trừu tượng của đoạn code đó trong bộ nhớ

Graphical user interface

Description automatically generated with medium confidence

HTML DOM cung cấp API để duyệt và chỉnh sửa các node. Nó chứa các phương thức như ***getElementById*** hay ***removeChild***.

A picture containing text

Description automatically generated

Chúng ta sử Javascript để tương tác với DOM. Tuy nhiên, sự tương tác thường không hiệu quả vì DOM được chủ ý riêng cho giao diện người dùng tĩnh. Các trang được trả về bởi Server nên không cần dùng cập nhập động. Khi DOM được cập nhập, nó cũng phải cập nhập node cũng như render lại trang cho phù hợp với CSS và cách bố trí.

Cùng với sự phát triển và phổ biến của SPA (Single Page App), các thành phần của trang web đang ngày càng có trách nhiệm hơn trong việc lắng nghe những cập nhập và render lại những thay đổi đó trên UI (giao diện người dùng). Không phải là hiếm khi gặp các trang hiển thị hàng nghìn các node được tạo ra tự động, mà vẫn phải tiếp tục lắng nghe các cập nhập mới trong tương lai. Đây là nơi mà mọi thứ có thể khá tốn kém.

Có một vài cách để các thành phần có thể biết khi nào một cập nhật dữ liệu xảy ra và có cần phải tạo lại giao diện người dùng hay không:

* **Dirty check** ( chậm) – Kiểm tra thông qua tất cả các node dữ liệu tại một khoảng thời gian đều đặn để xem có gì thay đổi hay không. Nó là không hiệu quả bởi vì nó yêu cầu đi qua các từng node đệ quy để đảm bảo rằng dữ liệu không bị “dirty” (hết hạn).
* **Observable**(nhanh) – Các thành phần sẽ chịu trách nhiệm lắng nghe khi một thay đổi diễn ra. Sau đó dữ liệu được lưu trên trạng thái, các thành phần đơn giản là lắng nghe các sự kiện trên trạng thái và nếu nó là một thay đổi thì nó có thể render lại UI.

Chúng ta có ví dụ về danh sách các item. Trong danh sách này, chúng ta có một số các item và một item bất kì cũng có thể nhận được yêu cầu cập nhập.

Table

Description automatically generated

Nếu một trong những item này thay đổi, thì DOM sẽ render lại toàn bộ, đây chính là lý do khởi nguồn của DOM không hiệu quả.

Table

Description automatically generated

Trong khi đây là một ví dụ đơn giản của một danh sach chứa số lượng ít các item, nó sẽ trở nên vô cùng không hiệu quả nêý chúng ta có một SPA (single page application) với hàng trăm, hàng nghìn các thành phần mà đòi hỏi render lại khi các cập nhập được truyền xuống. Lý tưởng nhất là chúng ta chỉ render lại các item mà nhận được cập nhập, bỏ qua các item còn lại như cách React sử dụng Virtual DOM giúp giảm việc làm không hiệu quả trên.

Vậy Virtual DOM là gì? Virtual DOM là một abstraction nhẹ của DOM. Chúng ta có thể coi nó như là một bản sao của DOM, mà việc cập nhập sẽ không gây ảnh hưởng tới DOM thực. Nó có tất cả các thuộc tính giống như object DOM thật, nhưng nó không có khả năng viết lên màn hình như DOM thật. Virtual DOM mới sẽ được tạo sau khi render lại. Lưu ý quan trọng là React không giới thiệu Virtual DOM vì có một vài thư viện cũng được tạo bằng cách sử dụng nó. Trong thực tế, khi mà React đang chuyển sang một số nền tảng không phải web như React-Native, React-VR,…. Phần từ các phiên bản React trước đây dùng để xử lý tương tác với DOM đã được mang ra. Nó giờ đây được cung cấp qua các package riêng biệt. Chúng ta cần phải cài đặt và bao hàm React Dom nếu muốn sử dụng React trên web.

React nắm tất cả các bảo sao của Virtual DOM (các bản sao ghi lại trạng thái DOM) trước khi áp dụng bất kỳ cập nhập nào. Nó sau đấy sử dụng bảo sao này so sánh với những thay đổi của Virtual DOM trước khi tạo ra các thay đổi.

Background pattern

Description automatically generated

Khi các thay đổi được đưa vào Virtual DOM, React sẽ sử dụng một quá trình được gọi là **reconciliation** (hoà hợp) – sử dụng một “**diffing**” thuật toán để so sánh và đối chiếu lại những thay đổi mục đích để biết được cập nhập nào đang diễn ra. React sau đấy chỉ cập nhập những phần tử đã thay đổi, bỏ qua những chỗ khác (không thay đổi)

Diagram, table

Description automatically generated

Diagram

Description automatically generated

### **1.4.3. Performance**

React sử dụng VDOM, giúp các ứng dụng web chạy nhanh hơn nhiều so với việc phát triển bằng các framework khác thay thế. React chia giao diện người dùng phức tạp thành các thành phần riêng lẻ, cho phép người sử dụng làm việc trên từng thành phần cùng lúc, dó đó đẩy nhanh thời gian phát triển ứng dụng.

### **1.4.4. Extensions**

React vượt xa thiết kế UI đơn giản và có rất nhiều extension cung cấp hỡn trợ kiến trúc ứng dụng một cách hoàn chỉnh. React cung cấp server-side rendering, đòi hỏi hiển thị một ứng dụng web thông thường chỉ phía bên server, sau đó gửi một trang đã render đầy đủ đến phía client. React cũng sử dụng Flux và Redux một cách chuyên sâu trong việc phát triển web application. Cuối cùng, chúng ta có React-Native, một framework phổ biến có nguồn gốc từ React, được sử dụng để tạo ra các ứng dụng mobile có tính tương thích chéo.

### **1.4.5. One-way data binding**

React sử dụng luồng dữ liệu một chiều, buộc các nhà phát triển phải sử dụng tính năng gọi lại để chỉnh sửa các thành phần và ngăn họ chỉnh sửa trực tiếp. Việc kiểm soát luồng dữ liệu từ một thời điểm duy nhất cung cấp bởi component có kiến trúc javascript app được gọi là Flux. Nó thực sự giúp các nhà phát triển kiểm soát tốt hơn toàn thể ứng dụng và làm cho nó trở nên linh hoạt và hiệu quả.

### **1.4.6. Component Based Architecture**

Điều này đơn giản có nghĩa là giao diện người dùng của một app dựa trên ReactJS được nhiều thành phần gộp lại, với mỗi thành phần có logic cụ thể, được viết bằng Javascript. Do đó, các nhà phát triển có thể chuyển tiếp dữ liệu trên toàn ứng dụng mà không làm hưởng đến DOM. Các component của ReactJS đóng một vai trò lớn trong việc quyết định cách app được nhìn và tương tác như thế nào.

## **1.5. So sánh giữa React với Vue.Js và Angular**

Hiện nay có thể lập trình front-end với bất kì ngôn ngữ lập trình nào. Tuy nhiên, những mã này phải được chuyển đổi thành Javascript vì browser hiện nay chỉ thực thi được các mã Javascript. Sự ra đời của WebAssembly, một trong những xu hướng phát triển web chính có thể thay đổi tình hình hiện nay. Tuy nhiên Javascript vẫn là một lựa chọn tối ưu để phát triển front-end. Có rất nhiều Javascript framework hiện nay khiến cho việc phát triển client-side dễ dàng và nhanh chóng hơn. Chúng ta sẽ đề cập đến ba trong số những framework phổ biến và có hiệu suất cao nhất hiện nay thường được chọn và sử dụng trong các project cùng với React: Angular, Vue.Js và Ember.

### **1.5.1. Angular**

Angular được phát hành lần đầu tiên vào năm 2010 bởi Google. Năm 2016, Angular 2 được công bố, là bản được viết lại hoàn toàn của Angular. Kể từ đó, phiên bản mới này xuất hiện thường xuyên. Angular 13 là phiên bản stable mới nhất hiện nay tính đến tháng 12 năm 2021.

**Ưu điểm:**

* MVC framework: Angular là một framework MVW (Model-View-Whatever), thông thường được sử dụng như một MVC framework. Do đó, ứng dụng được chia thành ba thành phần kết nối với nhau. Điều này cho phép các nhà phát triển sử dụng Angular JS có thể viết những ứng dụng với cấu trúc rất rõ ràng, điều này có lợi cho các project phức tạp.
* Angular templates: template để tạo ra các component dễ đọc do hầu hết chúng sử dụng những thẻ tag HTML cơ bản.
* Thực thi dễ dàng với liên kết dữ liệu hai chiều: nghĩa là bất kỳ thay đổi nào với model đều ảnh hưởng đến view. Ngược lại nếu view thay đổi thì model cũng sẽ thay đổi theo. Angular cho phép liên kết dữ liệu hai chiều đơn giản, có lợi cho các ứng dụng đơn giản. Các ứng dụng phức tạp hơn hoạt động nhanh hơn với liên kết dữ liệu một chiều (view-to-model hoặc model-to-view), dựa trên nhu cầu của ứng dụng. Điều này cho phép tiết kiệm thêm tài nguyên.
* Cộng đồng lớn: Angular có một cộng đồng lớn đã phát tiển kể từ khi phát hành Angular JS và ngày càng lớn mạnh hơn kể từ khi Angular 2 được phát hành. Framework này đạt gần dến 500.000 lượt tải mỗi tuần trên npm và hơn 45.000 sao trên Github. Sự phổ biến này có nghĩa là có một số giải pháp tương thích với các phiên bản khác nhau của Angular, cũng như khả năng nhận lời khuyên từ các nhà phát triển và người dùng có nhiều kinh nghiệm mà không cần phải liên hệ với đội ngũ hỗ trợ chính thức để có đáp án cho vấn đề được đề cập.

**Nhược điểm:**

* Không sử dụng shadow DOM là mặc định: shadow DOM giải quyết vấn đề về tên độc nhất cho phần tử của trang hoặc ID, điều mà có thể tạo ra cho các nhà phát triển nhiều sự phiền toái, đặc biệt nếu nó là một project phức tạp. Một nhà phát triển có thể thay đổi style CSS trong một file nào đó và một file khác có thể bị ảnh hưởng theo. Shadow DOM cho phép browser chứa thêm một cây con bao gồm các phần tử DOM trong việc render nhưng không chứ trong DOM chính. Shadow DOM đóng gói các kiểu, tập lệnh và nội dung bên trong một phần tử tùy chỉnh để chúng không ảnh hưởng đến nội dung khác trong ứng dụng. Trong trường hợp của Angular 2+, shadow DOM được kích hoạt trong các trình duyệt hỗ trợ nó. Nếu không, quá trình đóng gói Shadow DOM được mô phỏng.
* Không sử dụng Virtual DOM: virtual DOM là một bản sao đơn giản của DOM. Virtual DOM cho phép thay đổi bất kỳ một phần tử nào một cách nhanh chóng mà không cần phải render lại toàn bộ DOM. Cách tiếp cận này là một xu hướng của các framework phát triển web hiện đại nhưng không may, nó không được hỗ trợ trong bất kỳ phiên bản nào của Angular. Thay vào đó, Angular 2 sử dụng luồng dữ liệu một chiều để phát hiện các thay đổi của model và chỉ cập nhập lại các phần bị ảnh hưởng khi model bị thay đổi.
* Sử dụng Typescript: là một Javascript framework, Angular hổ trợ sử dụng Javascript thuần túy. Tuy nhiên framework này được tạo ra để sử dụng với Typescript, một phiên bản nâng cấp của Javascript được tạo bởi Mycrosoft. Typescript cho phép sức mạnh thực sự của Angular thật sự được giải phóng. Có một điều bất lợi là chúng ta cần phải học lại các cú pháp đã được sửa đổi.
* Tốc độ render thấp: việc giới hạn sử dụng shadow DOM và thiếu sử dụng Virtual DOM dẫn đến việc hiệu suất thấp. Việc render/re-render view tốc độ thấp hơn khi so sánh với các framework Javascript khác được so sánh.
* Code cần nhiều tài nguyên: Angular là một monolithic framework với mã nguồn nặng cần được tải xuống từ server trước khi chúng ta thấy được web app trên browser. Kết quả là tốc độ và hiệu suất giảm đi. May mắn thay, những vấn đề này có thể giải quyết bằng cách sử dụng tree-shaking, một kỹ thuật cho phép loại bỏ các đoạn code không sử dụng trong ứng dụng. Webpack là môi trường hoàn hảo cho tree-shaking. Sau đây là các nó hoạt động: webpack quét các đoạn code, đặt tất cả các module trong một file, và xóa tất cả export từ code mà không được export. Kết quả là việc đóng gói sẽ loại bỏ các mã code không sử dụng.

**Những điều thay đổi của Angular 13:**

* Typescript 4.4 được hỗ trợ; kể từ ngày 5 tháng 10, những phiên bản cũ hơn 4.4.2 sẽ không còn được hỗ trợ ở core.
* Phiên bản Node.js cũ hơn v12.20 không còn được hỗ trợ, do Angular package sử dụng chức năng export của Node.js với subpath patterns.
* Thư viện lập trình rxjs (Reactive Extensions for Javascript) v7 được hỗ trợ.
* Validator built-in như min, max, và minLength có được được disable hoặc enable một cách linh động.
* Một API đơn giản ViewContainerRef.createComponent cho phép tạo các component một cách linh hoạt.
* Thông báo lỗi được cải thiện.
* Khả năng “build cache by default” được cải thiện chất lượng build.
* Angular Package Format 13 được giới thiệu, xóa bỏ View Engine-specific code từ package. APF được sắp xếp hợp lý và hiện đại hóa.
* $locallize là một stable API.
* Điều kiện tùy chỉnh có thể được cài đặt trong ng\_package.
* Router không thay thế URL của browser khi điều hướng mới hủy đi điều hướng đang hoạt động.
* Lớp wrappedValue không còn được hỗ trợ trong @angular/core.
* Behavior của SpyLocation sử dụng bở RouterTestingModule đã bị thay đổi để phù hợp với behavior của browser.

### **1.5.2. Vue.js**

Vue.js là một sự thay thế phổ biến của Angular và React. Framework hiện đại này dành cho build UI ngày càng trở nên phổ biến. Đầu tiên nó cực kỳ phổ biến ở trung quốc và bây giờ nó cũng phổ biến ở phương tây.

**Ưu điểm:**

* MVC framework: cũng như Angular, Vue là một MVC framework. Ưu điểm của điều này là cho phép chúng ta viết những đoạn code có cấu trúc tốt, điều mà rất quan trong trong việc phát triển những app phức tạp.
* Giải pháp gọn nhẹ: một lợi thế lớn của Vue.js là kích thước framework nhỏ vì không bao gồm nhiều tính năng thừa, nhưng chức năng dễ dàng mở rộng với nhiều giải pháp của các bên thứ ba. Nó thường được so sánh với Angular, là một monolithic framework có một loạt các tính năng được tích hợp sẵn mà hầu như không sử dụng đến trong ứng dụng được phát triển. Dĩ nhiên tree-shaking vẫn cho phép chúng ta triệt tiêu đi những code không sử dụng nhưng kích thước của framework vẫn là rất lớn so sánh với những gì mà Vue đem lại.
* Declarative template: Template trong Vue.js được viết bằng HTML, điều đó làm nó dễ dàng đọc mà không cần kiến thức về ngôn ngữ lập trình khác.
* Virtual DOM: Do sử dụng virtual DOM, ứng dụng được built với Vue.js có hiệu suất cao khi so sánh với những framework front-end khác.
* Liên kết dữ liệu hai chiều: Vue.js tự động đồng bộ hóa tất cả model với DOM.
* Hỗ trợ Laravel: Laravel là một trong những PHP back-end framework tốt nhất. Hỗ trợ của Vue.js cho phép web app được xây dựng sử dụng 2 trong số các công nghệ tiên tiến nhất mà không cần thêm các cài đặt hỗ trợ.
* Javascript thuần khiết: Vue.js sử dụng Javascript thuần khiết, loại bỏ yêu cầu đối với nhà phát triển hoặc tester để học thêm bất cứ ngôn ngữ nào khác.
* Càng ngày phổ biến:với gần 800.000 lượt tải về, Vue.js có một kết quả khả quan.Hơn 128.000 sao trên github thể hiện mức độ phổ biến và phản ánh quan điểm của cộng đồng. Vue.js đang được sử dụng trên khắp thế giới. Điều này có nghĩa các giải pháp tương thích với Javascript framework này sẽ càng trở nên sẵn có.
* Learning curve thấp: so sánh với Angular và React, Vue.js là công nghệ front-end dễ học và bắt đầu nhất.Thêm vào đó, chúng ta có thể bắt đầu thêm Vue.js vào project từng bước một.

**Nhược điểm:**

* Cộng đồng nhỏ: Vue ít phổ biến hơn khi so sánh với Angular và React, cả hai đều có một lượng người dùng ấn tượng. Tuy nhiên, chúng ta có thể dựa vào sự hỗ trợ của các thành viên trong cộng đồng.
* Ít thư viện hơn dành cho Vue.js: với việc ít người dùng hơn, có ít giải pháp được đưa ra để mở rộng thêm tính năng cho framework.
* Nguồn gốc trung quốc: framework này được tạo bởi Evan You, một cựu lập trình viên Angular. Do đó, ông hiểu điểm yếu của Angular ở đâu và có khả năng xây dựng một giải pháp thay thế giải quyết các vấn đề của Angular trong khi vẫn giữ lại những điểm mạnh của nó. Framework mới này nhanh chóng trở nên cực kì phổ biến tại đất nước của ông, Trung quốc. Kết quả là, tài liệu dành cho một số thư viện thứ ba chỉ khả dụng với tiếng trung. Nhưng tài liệu chính thức đã hoàn toàn có sẵn bằng tiếng anh.

**Khi nào nên dùng Vue.js:** Vue có thể được sử dụng trong nhiều loại application khác nhau. Vì sự tương thích của nó với các thư viên Javascript khác và khả năng thêm các logic phức tạp vào trong app đã tồn tại trước đó, Vue có thể cung cấp một giải pháp hoàn hảo cho bất kỳ project nào, Tuy nhiên, chúng ta nên xem xét Vue.js như là một lựa chọn chính nếu chúng ta cần các yêu cầu dưới đây:

* Application năng động hiệu suất cao: Tương tự như React, Vue.js là một lựa chọn tốt dành cho các application động, tuy nhiên nhờ vào Virtual DOM, cung cấp hiệu suất cao, điều này thì có lợi cho các app phức tạp.
* Single page app: Vue.js cho phép nội dung thay đổi nhanh chóng mà không cần tải lại trang, điều này làm cho nó hoàn hảo để phát triển SPAs.

### **1.5.3. EmberJS**

Ember 3.13 ra mắt năm vừa qua với một số những cập nhật và tính năng mới. Ember giống như Backbone và Angular JS, và cũng là một trong những Javascript framework lâu đời. Với bản cập nhật mới, Ember 3.13 tương thích với các bản fix bug, cải thiện hiệu suất và phản hồi. Các bản cập nhật thuộc tính được theo dõi cũng đưa ra những cách thức đơn giản hơn để đo lường sự thay đổi state trong hệ thống công thái học (ergonomic system) của các ứng dụng Ember.

Ember có kiến trúc tương đối phức tạp, cho phép bạn nhanh chóng dựng được các bản ứng dụng lớn phía client (client-side apps) . Ember thực hiện chức năng của một JavaScript frameworks MVC điển hình và kiến trúc của Ember bao gồm các thành phần sau: adapter, component, controller, helper, model, route, service, template, util và addon.

Một trong những tính năng tốt nhất của Ember là công cụ giao diện dòng lệnh (command line interface tool). Ember CLI hỗ trợ các lập trình viên front-end nâng cao năng suất và giúp họ hoàn thiện dự án của mình đúng thời hạn. Bạn không chỉ đơn thuần hoàn thành các dự án mới nhờ những bộ cài đặt sẵn mà còn có thể tạo ra các controller, component và các file dự án thông qua các thế hệ tự động.

Hệ sinh thái EmberJS bao gồm:

* Công cụ Ember CLI (Ember CLI tool) dành cho việc tạo nguyên mẫu (prototype) nhanh và quản lý các phần phụ.
* Xây dựng máy chủ Ember trong framework khi lập trình, phát triển các ứng dụng.
* Thư viện Ember.js và Ember Data trong quản lý dữ liệu.
* Công cụ tạo mẫu (template engine) Handlebars với các ứng dụng Ember.
* Framework kiểm thử QUnit cho Ember
* Công cụ phát triển Ember Inspector cho các trình duyệt Chrome và Firefox
* Ember Observer cho các kho công cộng và Ember addon để thực hiện các chức năng di truyền.

## **1.6. Quy mô cộng đồng phát triển:**

Các thành viên trong nhóm React làm việc toàn thời gian trên các API core component, engine hỗ trợ React DOM và React Native, React DevTools và tài liệu trên website React.

Các thành viên trong React team được liệt kê theo thứ tự bảng chữ cái bên dưới:

**Andrew Clark**



**Brian Vaughn**



**Dan Abramov**



**Luna Ruan**



**Macro Salazar**



**Rachel Nabors**



**Rick Halon**



**Sebastian Markbåge**



**Seth Webster**



## **1.7. Quy mô cộng đồng sử dụng:**

React có cộng đồng người sử dụng cực lớn khi so sánh với các framework khác như Angular hay Vue.js. Hàng triệu lập trình viên tin dùng và gần 1.400 lập trình viên đóng góp vào việc phát triển React.

Graphical user interface, chart, application, line chart

Description automatically generated

Biểu đồ npm trend 5 năm gần đây

Chart, line chart

Description automatically generated

Google trend theo thời gian của React

Graphical user interface, text, application

Description automatically generated

Google trend theo khu vực của React

A screenshot of a computer

Description automatically generated with medium confidence

Repo của React đạt 180.000 sao

## **1.8. Kiến trúc**

ReactJS là một thư viên của Facebook để lập trình client-side web app. React Native là một thư viện khác với cấu trúc gần như giống hệt React nhưng để lập trình client-side mobile app, và có thể chạy trên cả Android lẫn iOS. Các khái niệm về ReactJS và React Native đều được gọi ngắn gọn là React, vì cũng có thể áp dụng được cho React Native.

React áp dụng một cấu trúc cây nhà lá vườn của Facebook: Flux. Khác với cấu trúc của một số framework khác (vd: AngularJS) sử dụng bidirectional data flow (dữ liệu đi 2 chiều) thì data trong Flux luôn đi một chiều unidirectional.

Các app React rất thường sử dụng thêm một thư viện quản lý global state management Redux, là một thư viện có vẻ lấy tên từ “reducer” + “flux”. Đại khái là thư viện này giúp thực hiện việc “reduce” (giải thích ở dưới) trong cấu trúc flux.

Cần nhấn mạnh là React không nhất thiết phải dùng Redux, mà có thể dùng thư viện khác để quản lý global state (vd: MobX) hoặc không cần quản lí global state luôn. Redux cũng là một thư viện độc lập không phải của Facebook, hoàn toàn có thể áp dụng cho các framework khác ngoài React. Chỉ là hai thứ này rất hợp nhau nên thường xuất hiện cùng nhau.

Redux có cấu trúc rất đơn giản: chứa một global state, mỗi khi nhận action thì trả về một state mới tùy theo nội dùng của action đó.

Để thực hiện thêm các tính năng phức tạp hơn thì ta dùng các middleware: trong ví dụ này là Redux-persist, Redux-saga, và Redux-logger.

Diagram

Description automatically generated

# **2. ExpressJS**

## **2.1. ExpressJs là gì**

ExpressJS là một framework ứng dụng web có mã nguồn mở và miễn phí được xây dựng trên nền tảng Node.js. ExpressJS được sử dụng để thiết kế và phát triển các ứng dụng web một cách nhanh chóng. Để hiểu ExpressJS, người dùng chỉ cần phải biết Javascript, do đó nên việc xây dựng các ứng dụng web và API trở nên đơn giản hơn đối với các lập trình viên và nhà phát triển đã thành thạo JavaScript trước đó.

Vì ExpressJS là một framework của Node.js nên hầu hết các mã đã được viết sẵn cho các lập trình viên làm việc. Chúng ta có thể tạo các ứng dụng web cho một trang, nhiều trang hoặc kết hợp lại bằng cách sử dụng ExpressJS. framework này khá nhẹ, giúp tổ chức các ứng dụng web ở phía máy chủ thành một kiến ​​trúc MVC hoàn hảo hơn.

ExpressJS hỗ trợ nâng cao các chức năng của NodeJS. Nếu chúng ta không sử dụng ExpressJS, chúng ta phải thực hiện rất nhiều lập trình phức tạp để xây dựng một API hiệu quả. ExpressJS đã giúp cho việc lập trình trong NodeJS trở nên dễ dàng hơn rất nhiều.

## **2.2. ExpressJs được dùng để làm gì**

ExpressJS sẽ giúp chúng ta tổ chức kiến ​​trúc back-end của mình. Các nhà phát triển web thường sử dụng ExpressJS để triển khai kiến ​​trúc MVC, điều này cho phép họ viết một codebase back-end bảo trì tương đối dễ dàng.

Bởi vì ExpressJS hoạt động trên back-end, nên chúng ta có thể coi công nghệ này như một “bộ não đằng sau một trang web”. Ví dụ: ExpressJS có thể xác định cách những trang được định tuyến trên một trang web. Hơn nữa, một nhà phát triển có thể sử dụng ExpressJS để quản lý xác thực trên một trang web.

Dưới đây là một số trường hợp sử dụng ExpressJs:

* Sử dụng cookie trên một trang web.
* Triển khai xác thực.
* Thêm thanh tìm kiếm vào một trang web.
* Cung cấp các tệp tĩnh như hình ảnh.

## **2.3. Lược sử ra đời**

ExpressJs được phát triển bởi TJ Holowaychuk. Bản phát hành đầu tiên, dựa theo lịch sử repo trên github của ExpressJS, là vào ngày 22 tháng 5 năm 2010, version 0.12.

Vào tháng 6 năm 2014, quyền quản lí dự án đã được StrongLoop mua lại. StrongLoop được IBM mua lại vào tháng 9 năm 2015. Vào tháng 1 năm 2016, IBM thông báo họ sẽ đặt ExpressJS dưới sự quản lí của chương trình Node.JS foundation.

**Release Change Log**

**4.17.2 - Release date: 2021-12-16**

The 4.17.2 patch release includes the following bug fixes:

* Fix handling of undefined in res.jsonp when a callback is provided.
* Fix handling of undefined in res.json and res.jsonp when "json escape" is enabled.
* Fix handling of invalid values to the maxAge option of res.cookie().
* Update to [jshttp/proxy-addr module](https://www.npmjs.com/package/proxy-addr) to use req.socket over deprecated req.connection.
* Starting with this version, Express supports Node.js 14.x.

For a complete list of changes in this release, see [History.md](https://github.com/expressjs/express/blob/master/History.md#4172--2021-12-16).

**4.17.1 - Release date: 2019-05-25**

The 4.17.1 patch release includes one bug fix:

* The change to the res.status() API has been reverted due to causing regressions in existing Express 4 applications.

For a complete list of changes in this release, see [History.md](https://github.com/expressjs/express/blob/master/History.md#4171--2019-05-25).

**4.17.0 - Release date: 2019-05-16**

The 4.17.0 minor release includes bug fixes and some new features, including:

* The express.raw() and express.text() middleware have been added to provide request body parsing for more raw request payloads. This uses the [expressjs/body-parser module](https://www.npmjs.com/package/body-parser) module underneath, so apps that are currently requiring the module separately can switch to the built-in parsers.
* The res.cookie() API now supports the "none" value for the sameSite option.
* When the "trust proxy" setting is enabled, the req.hostname now supports multiple X-Forwarded-For headers in a request.
* Starting with this version, Express supports Node.js 10.x and 12.x.
* The res.sendFile() API now provides and more immediate and easier to understand error when a non-string is passed as the path argument.
* The res.status() API now provides and more immediate and easier to understand error when null or undefined is passed as the argument.

For a complete list of changes in this release, see [History.md](https://github.com/expressjs/express/blob/master/History.md#4170--2019-05-16).

**4.16.4 - Release date: 2018-10-10**

The 4.16.4 patch release includes various bug fixes:

* Fix issue where "Request aborted" may be logged in res.sendfile.

For a complete list of changes in this release, see [History.md](https://github.com/expressjs/express/blob/master/History.md#4164--2018-10-10).

4.16.3 - Release date: 2018-03-12

The 4.16.3 patch release includes various bug fixes:

* Fix issue where a plain % at the end of the url in the res.location method or the res.redirect method would not get encoded as %25.
* Fix issue where a blank req.url value can result in a thrown error within the default 404 handling.
* Fix the generated HTML document for express.static redirect responses to properly include </html>.

For a complete list of changes in this release, see [History.md](https://github.com/expressjs/express/blob/master/History.md#4163--2018-03-12).

**4.16.2 - Release date: 2017-10-09**

The 4.16.2 patch release includes a regression bug fix:

* Fix a TypeError that can occur in the res.send method when a Buffer is passed to res.send and the ETag header is already set on the response.

For a complete list of changes in this release, see [History.md](https://github.com/expressjs/express/blob/master/History.md#4162--2017-10-09).

4.16.1 - Release date: 2017-09-29

The 4.16.1 patch release includes a regression bug fix:

* Update to [pillarjs/send module](https://www.npmjs.com/package/send) to fix an edge case scenario regression that affected certain users of express.static.

For a complete list of changes in this release, see [History.md](https://github.com/expressjs/express/blob/master/History.md#4161--2017-09-29).

**4.16.0 - Release date: 2017-09-28**

The 4.16.0 minor release includes security updates, bug fixes, performance enhancements, and some new features, including:

* Update to [jshttp/forwarded module](https://www.npmjs.com/package/forwarded) to address a [vulnerability](https://npmjs.com/advisories/527). This may affect your application if the following APIs are used: req.host, req.hostname, req.ip, req.ips, req.protocol.
* Update a dependency of the [pillarjs/send module](https://www.npmjs.com/package/send) to address a [vulnerability](https://npmjs.com/advisories/535) in the mime dependency. This may affect your application if untrusted string input is passed to the following APIs: res.type().
* The [pillarjs/send module](https://www.npmjs.com/package/send) has implemented a protection against the Node.js 8.5.0 [vulnerability](https://nodejs.org/en/blog/vulnerability/september-2017-path-validation/). Using any prior version of Express.js with Node.js 8.5.0 (that specific Node.js version) will make the following APIs vulnerable: express.static, res.sendfile, and res.sendFile.
* Starting with this version, Express supports Node.js 8.x.
* The new setting "json escape" can be enabled to escape characters in res.json(), res.jsonp() and res.send() responses that can trigger clients to sniff the response as HTML instead of honoring the Content-Type. This can help protect an Express app from a class of persistent XSS-based attacks.
* The [res.download() method](https://expressjs.com/en/4x/api.html#res.download) now accepts an optional options object.
* The express.json() and express.urlencoded() middleware have been added to provide request body parsing support out-of-the-box. This uses the [expressjs/body-parser module](https://www.npmjs.com/package/body-parser) module underneath, so apps that are currently requiring the module separately can switch to the built-in parsers.
* The [express.static() middleware](https://expressjs.com/en/4x/api.html#express.static) and [res.sendFile() method](https://expressjs.com/en/4x/api.html#res.sendFile) now support setting the immutable directive on the Cache-Control header. Setting this header with an appropriate maxAge will prevent supporting web browsers from sending any request to the server when the file is still in their cache.
* The [pillarjs/send module](https://www.npmjs.com/package/send) has an updated list of MIME types to better set the Content-Type of more files. There are 70 new types for file extensions.

For a complete list of changes in this release, see [History.md](https://github.com/expressjs/express/blob/master/History.md#4160--2017-09-28).

**4.15.5 - Release date: 2017-09-24**

The 4.15.5 patch release includes security updates, some minor performance enhancements, and a bug fix:

* Update to [debug module](https://www.npmjs.com/package/debug) to address a [vulnerability](https://snyk.io/vuln/npm:debug:20170905), but this issue does not impact Express.
* Update to [jshttp/fresh module](https://www.npmjs.com/package/fresh) to address a [vulnerability](https://npmjs.com/advisories/526). This will affect your application if the following APIs are used: express.static, req.fresh, res.json, res.jsonp, res.send, res.sendfile res.sendFile, res.sendStatus.
* Update to [jshttp/fresh module](https://www.npmjs.com/package/fresh) fixes handling of modified headers with invalid dates and makes parsing conditional headers (like If-None-Match) faster.

For a complete list of changes in this release, see [History.md](https://github.com/expressjs/express/blob/master/History.md#4155--2017-09-24).

**4.15.4 - Release date: 2017-08-06**

The 4.15.4 patch release includes some minor bug fixes:

* Fix array being set for "trust proxy" value being manipulated in certain conditions.

For a complete list of changes in this release, see [History.md](https://github.com/expressjs/express/blob/master/History.md#4154--2017-08-06).

**4.15.3 - Release date: 2017-05-16**

The 4.15.3 patch release includes a security update and some minor bug fixes:

* Update a dependency of the [pillarjs/send module](https://www.npmjs.com/package/send) to address a [vulnerability](https://snyk.io/vuln/npm:ms:20170412). This may affect your application if untrusted string input is passed to the maxAge option in the following APIs: express.static, res.sendfile, and res.sendFile.
* Fix error when res.set cannot add charset to Content-Type.
* Fix missing </html> in HTML document.

For a complete list of changes in this release, see [History.md](https://github.com/expressjs/express/blob/master/History.md#4153--2017-05-16).

**4.15.2 - Release date: 2017-03-06**

The 4.15.2 patch release includes a minor bug fix:

* Fix regression parsing keys starting with [ in the extended (default) query parser.

For a complete list of changes in this release, see [History.md](https://github.com/expressjs/express/blob/master/History.md#4152--2017-03-06).

**4.15.1 - Release date: 2017-03-05**

The 4.15.1 patch release includes a minor bug fix:

* Fix compatibility issue when using the datejs 1.x library where the [express.static() middleware](https://expressjs.com/en/4x/api.html#express.static) and [res.sendFile() method](https://expressjs.com/en/4x/api.html#res.sendFile) would incorrectly respond with 412 Precondition Failed.

For a complete list of changes in this release, see [History.md](https://github.com/expressjs/express/blob/master/History.md#4151--2017-03-05).

4.15.0 - Release date: 2017-03-01

The 4.15.0 minor release includes bug fixes, performance improvements, and other minor feature additions, including:

* Starting with this version, Express supports Node.js 7.x.
* The [express.static() middleware](https://expressjs.com/en/4x/api.html#express.static) and [res.sendFile() method](https://expressjs.com/en/4x/api.html#res.sendFile) now support the If-Match and If-Unmodified-Since request headers.
* Update to [jshttp/etag module](https://www.npmjs.com/package/etag) to generate the default ETags for responses which work when Node.js has [FIPS-compliant crypto enabled](https://nodejs.org/dist/latest/docs/api/cli.html#cli_enable_fips).
* Various auto-generated HTML responses like the default not found and error handlers will respond with complete HTML 5 documents and additional security headers.

For a complete list of changes in this release, see [History.md](https://github.com/expressjs/express/blob/master/History.md#4150--2017-03-01).

**4.14.1 - Release date: 2017-01-28**

The 4.14.1 patch release includes bug fixes and performance improvements, including:

* Update to [pillarjs/finalhandler module](https://www.npmjs.com/package/finalhandler) fixes an exception when Express handles an Error object which has a headers property that is not an object.

For a complete list of changes in this release, see [History.md](https://github.com/expressjs/express/blob/master/History.md#4141--2017-01-28).

**4.14.0 - Release date: 2016-06-16**

The 4.14.0 minor release includes bug fixes, security update, performance improvements, and other minor feature additions, including:

* Starting with this version, Express supports Node.js 6.x.
* Update to [jshttp/negotiator module](https://www.npmjs.com/package/negotiator) fixes a [regular expression denial of service vulnerability](https://npmjs.com/advisories/106).
* The [res.sendFile() method](https://expressjs.com/en/4x/api.html#res.sendFile) now accepts two new options: acceptRanges and cacheControl.
  + acceptRanges (defaut is true), enables or disables accepting ranged requests. When disabled, the response does not send the Accept-Ranges header and ignores the contents of the Range request header.
  + cacheControl, (default is true), enables or disables the Cache-Control response header. Disabling it will ignore the maxAge option.
  + res.sendFile has also been updated to handle Range header and redirections better.
* The [res.location() method](https://expressjs.com/en/4x/api.html#res.location) and [res.redirect() method](https://expressjs.com/en/4x/api.html#res.redirect) will now URL-encode the URL string, if it is not already encoded.
* The performance of the [res.json() method](https://expressjs.com/en/4x/api.html#res.json) and [res.jsonp() method](https://expressjs.com/en/4x/api.html#res.jsonp) have been improved in the common cases.
* The [jshttp/cookie module](https://www.npmjs.com/package/cookie) (in addition to a number of other improvements) has been updated and now the [res.cookie() method](https://expressjs.com/en/4x/api.html#res.cookie) supports the sameSite option to let you specify the [SameSite cookie attribute](https://tools.ietf.org/html/draft-west-first-party-cookies-07). NOTE: This attribute has not yet been fully standardized, may change in the future, and many clients may ignore it.

The possible value for the sameSite option are:

* + true, which sets the SameSite attribute to Strict for strict same site enforcement.
  + false, which does not set the SameSite attribute.
  + 'lax', which sets the SameSite attribute to Lax for lax same site enforcement.
  + 'strict', which sets the SameSite attribute to Strict for strict same site enforcement.
* Absolute path checking on Windows, which was incorrect for some cases, has been fixed.
* IP address resolution with proxies has been greatly improved.
* The [req.range() method](https://expressjs.com/en/4x/api.html#req.range) options object now supports a combine option (false by default), which when true, combines overlapping and adjacent ranges and returns them as if they were specified that way in the header.

For a complete list of changes in this release, see [History.md](https://github.com/expressjs/express/blob/master/History.md#4140--2016-06-16).

## **2.4. Lợi ích khi sử dụng ExpressJS**

* Rất dễ học, chỉ cần chúng ta biết JavaScript, chúng ta sẽ không cần phải học một ngôn ngữ mới để học ExpressJS
* Giúp cho việc phát triển back-end dễ dàng hơn nhiều khi sử dụng ExpressJS
* Mã JavaScript được diễn giải thông qua Google V8 JavaScript Engine của Node.js. Do đó, mã sẽ được thực hiện một cách nhanh chóng và dễ dàng.
* ExpressJS rất đơn giản để tùy chỉnh và sử dụng theo nhu cầu.
* Cung cấp một module phần mềm trung gian linh hoạt và rất hữu ích để thực hiện các tác vụ bổ sung theo phản hồi và yêu cầu.
* Hỗ trợ phát triển ứng dụng theo mô hình MVC, đây là mô hình phổ biến cho việc lập trình web hiện nay.

## **2.5. Danh sách tính năng cơ bản**

* Phát triển máy chủ nhanh chóng: ExpressJS cung cấp nhiều tính năng thường được sử dụng của Node.JS dưới dạng các hàm có thể dễ dàng sử dụng ở bất kỳ đâu trong chương trình. Điều này loại bỏ nhu cầu viết mã trong vài giờ và tiết kiệm thời gian.
* Middleware: là một phần của chương trình có quyền truy cập vào cơ sở dữ liệu, yêu cầu của client và các phần mềm trung gian khác. Nó chịu trách nhiệm chính cho việc tổ chức có hệ thống các chức năng khác nhau của ExpressJS.
* Routing: ExpressJS cung cấp cơ chế định tuyến nâng cao giúp duy trì trạng thái của trang web với sự trợ giúp của URL.
* Templating: ExpressJS cung cấp các templating engines cho phép nhà phát triển có thể xây dựng nội dung linh động của trang web bằng cách xây dụng HTML template trên server-side.
* Debugging: Debbugging là rất quan trong trong việc phát triển thành công các ứng dụng web, ExpressJS giúp việc gỡ lỗi trở nên dễ dàng hơn bằng cách cung cấp các cơ chế gỡ lỗi có khả năng xác định chính xác phần ứng dụng web có lỗi.

## **2.6. Một số công nghệ tương đương**

### **2.6.1 Hapi.js**

Hapi.js là một REST API-based framework nhằm xây dựng các ứng dụng web có thể mở rộng và phát triển bởi team phát triển mobile của Walmart Labs. Một sự thật thú vị về Hapi.js là nó đã xử lý tất cả các yêu cầu của ứng dụng di động của Walmart vào dịp black Friday với khoảng 10 cpu và 28Gb ram.

**Tính năng:**

* Nó cung cấp các chức năng có thể tái sử dụng và lưu vào bộ nhớ cache mà chúng ta có thể truy cập trong toàn bộ ứng dụng.
* Nó cung cấp các lớp trừu tượng giúp chia ứng dụng thành nhiều phần hợp lý.
* Nó tuân theo phương pháp không conflic routing. Các đường dẫn sẽ không xảy ra hiện tượng conflic.

**Ưu điểm:**

* Nó cung cấp một hệ thống plugin mạnh mẽ và có thể mở rộng giúp phát triển ứng dụng nhanh hơn và hiệu quả hơn.
* Nó chủ yếu cung cấp các module như cơ chế bộ nhớ đệm, xác thực và xác thực đầu vào ngay lập tức.
* Nó kiểm soát tốt hơn việc xử lý các request

**Nhược điểm:**

* Không có quy tắc hoặc hướng dẫn nào cho việc cấu trúc codebase. Vì vậy, việc tuân thủ theo tiêu chuẩn tùy thuộc vào nhà phát triển. Nó có thể trở thành một vấn đề nếu không được tuân thủ nghiêm ngặt theo một tiêu chuẩn nào đó.
* Mặc dù nó đi kèm với các module được dựng sẵn nhưng không có cách nào để Scanfold. Các endpoint phải được tạo ra và kiểm tra một cách thủ công.
* Khi mà đã kết hợp chặt chẽ với một ứng dụng sử dụng Hapi.js, việc chuyển sang dùng một framework khác là một thách thức lớn.

### **2.6.2. Sails.js**

Việc xây dựng một ứng dụng tầm cỡ doanh nghiệp từ ban đầu rất khó khăn. Chúng ta cần bảo trì cấu trúc code và nguyên tắc thiết kế trong toàn công ty. Một framework như Sails.js giải quyết các vấn đề đó cho chúng ta. Nó là một MVC-based framework dành cho ứng dụng web dành cho các ứng dụng cấp doanh nghiệp.

Nó giống với kiến trúc MVC từ những framework như Ruby hay Rails. Nó tuân theo nguyên tắc “convention over configuration”. Nó giúp chúng ta xây dựng các ứng dụng theo thời gian thực một cách nhanh chóng.

**Tính năng:**

* Nó sử dụng những tính năng bên trong ExpressJS
* Nó cung cấp WebSockets để xây dựng các ứng dụng theo thời gian thực.
* Nó cung cấp ORM/ODM làm cho nó trở thành cơ sở dữ liệu bất khả thi.
* Sails.js có khả năng tạo code mạnh mẽ. Nó có một khái niệm được gọi là blueprintactions giúp việc tạo code một các hiệu quả.
* Nó ưu tiên các quy ước hơn là config.

**Ưu điểm:**

* Kiến trúc MVC-based. Vì vậy, nó tách các logic nghiệp vụ ra khỏi UI.
* Nó cung cấp miễn phí JSON API generation. Và chúng ta không cần thêm routing.
* Nó cho phép chúng ta lưu trữ dữ liệu ở bất kỳ đâu. Khi nó sử dụng waterline ORM/ODM, cơ sở dữ liệu của nó là bất khả thi.

**Nhược điểm:**

* Sails.js chậm với mức tiêu thụ bộ nhớ cao.
* Việc quản lý các asset tĩnh trên ứng dụng có thể trở nên vụng về.
* Đôi lúc, việc phát triển trên một ứng dụng có thể chậm và tiêu tốn thời gian. Vì vậy, cần có thời gian để khởi động trên máy chủ product.

### **2.6.3. Koa.js**

Koa.js là web framework mới nhất được thiết kế bởi team đằng sau Express. Nó gợi lên một câu hỏi vì sao cần thêm một framework mới. Koa.js thì nhỏ hơn và được đóng gói sẵn các chức năng không đồng bộ như mặc định. Chúng ta nhận thấy rằng Express có những hạn chế đối với callback. Vì thế, nhóm quyết định giải quyết vấn đề đó với Koa.js bằng cách cung cấp các tác vụ không đồng bộ như mặc địn.

**Tính năng:**

* Koa hiện đại, được đảm bảo cho tương lai, và xây dựng trên ES6, một phiên bản Javascript hiện đại để xây dựng các ứng dụng web dễ dàng hơn.
* Nó sử dụng ES6 generator để tạo các chức năng downstream hoặc upstream. Nó giúp chúng ta tránh được callback hell.
* Nó cung cấp một module catchall dựng sẵn để kiểm soát lỗi hiệu quả hơn.
* Nó sử dụng context object, là một gói chứa các đối tượng yêu cầu và phản hồi.

**Ưu điểm**

* Cực kỳ nhẹ chỉ với 550 dòng code
* Nó sử dụng một generator để tránh cái callback ngay từ lúc bắt đầu.

**Nhược điểm**

* Sử dụng generator khiến cho ứng dụng khó chuyển sang framework khác trong tương lai. Nó tạo ra kết nối chặt chẽ với framework.
* Koa có cộng đồng nhỏ khi so sánh với ExpressJS.
* Nó không tương thích với phần mềm trung gian kiểu express.

## **2.7. Cộng đồng phát triển và sử dụng**

Nhà sáng lập express.js, Ông TJ Holowaychuk đã nhượng quyền quản lý lại cho StrongLoop vào năm 2014. StrongLoop đã bán lại chi IBM năm 2015. Hiện nay, Express.js đang dưới sự quản lý của Node.js Foundation (Theo wikipedia).

Quy mô cộng đồng sử dụng Express.js

Graphical user interface

Description automatically generated

Graphical user interface

Description automatically generated

Theo Google Trend Graph

Graphical user interface, chart

Description automatically generated

Theo npm trends

# **3. Xây dựng ứng dụng minh họa**

[Liên kết](https://github.com/DangQuang98UIT/Do_an_1)