# LỜI CẢM ƠN

Trong thời gian làm đồ án tốt nghiệp, em đã nhận được nhiều sự giúp đỡ, đóng góp ý kiến và sự chỉ bảo nhiệt tình của thầy cô, bạn bè, cũng như các đồng nghiệp trong công ty.

Em xin gửi lời cảm ơn chân thành đến Th.s Nguyễn Hiền Trinh giảng viên Bộ môn Khoa học máy tính và Công nghệ đã tận tình hướng dẫn, chỉ bảo em trong suốt quá trình làm đồ án.

Em cũng xin gửi lời cảm ơn đến các thầy cô trong trường Đại học Công nghệ thông tin và Truyền thông nói chung và các thầy cô trong khoa CNTT nói riêng đã dạy em những kiến thức và tạo điều kiện giúp đỡ em trong quá trình học tập góp phần giúp em hoàn thành đồ án tốt nghiệp này.

Cuối cùng em xin cảm ơn bạn bè và các đồng nghiệp đã có những góp ý giúp em hoàn thiện đồ án tốt nghiệp!

*Thái Nguyên , ngày 05 tháng 03 năm 2020*

Sinh viên thực hiện

# LỜI MỞ ĐẦU

Hiện nay nhu cầu tìm phòng của mọi người, mọi tầng lớp là rất cao, từ sinh viên, người lao động, hay đến các công nhân viên chức, gia đình … Nhưng phương thức tìm kiếm phòng trọ thông thường, đi tìm từng ngõ ngách và từng khu phố để tìm số điện thoại chủ nhà rồi gọi để xem phòng lại tốn quá nhiều thời gian mà hiệu cao lại không được cao. Chúng ta kì vọng là ngồi tại nhà vẫn tìm được phòng như ý muốn với thời gian tìm kiếm nhanh chóng và giá cả cũng hợp lý. Giải pháp đề ra là thiết lập một hệ thống tổng hợp danh sách các phòng trọ với đầy đủ thông tin. Vì vậy trong lần báo cáo đồ án này em đã chọn đề tài: “**Xây dựng website tìm phòng trọ khu vực Hà Nội**”. Người dùng có thể tìm kiếm các phòng theo mong muốn về diện tích, giá cả, khu vực… một cách nhanh chóng và thuận lợi.

# CHƯƠNG I: CƠ SỞ LÝ THUYẾT

## Tìm hiều về Javascript

### Tổng quan về Javascript

Javascript theo phiên bản hiện hành là một ngôn ngữ thông dịch được phát triển từ các ý niệm nguyên mẫu. Ngôn ngữ này được dùng rộng rãi cho các trang web (phía client) cũng như máy chủ (server). Nó là một trong những ngôn ngữ hiếm hoi có thể xây dựng được cả hai phía server-side và client-side.

### 1.2 Lịch sử ra đời

JavaScript được tạo trong mười ngày bởi Brandan Eich, một nhân viên của Netscape, vào tháng 9 năm 1995. Được đặt tên đầu tiên là Mocha, tên của nó được đổi thành Mona rồi LiveScript trước khi thật sự trở thành JavaScript nổi tiếng như bây giờ. Phiên bản đầu tiên của ngôn ngữ này bị giới hạn độc quyền bởi Netscape và chỉ có các tính năng hạn chế, nhưng nó tiếp tục phát triển theo thời gian, nhờ một phần vào cộng đồng các lập trình viên đã liên tục làm việc với nó.

Trong năm 1996, JavaScript được chính thức đặt tên là ECMAScript. ECMAScript 2 phát hành năm 1998 và ECMAScript 3 tiếp tục ra mắt vào năm 1999. Nó liên tục phát triển thành JavaScript ngày nay, giờ đã hoạt động trên khắp mọi trình duyệt và trên khắp các thiết bị từ di động đến máy tính bàn.

JavaScript liên tục phát triển kể từ đó, có lục đạt đến 92% website đang sử dụng JavaScript vào năm 2016. Chỉ trong 20 năm, nó từ một ngôn ngữ lập trình riêng trở thành công cụ quan trọng nhất trên bộ công cụ của các chuyên viên lập trình web. Nếu bạn đang dùng internet, vậy chắc chắn bạn đã từng sử dụng JavaScript rồi.

### 1.3 Ưu, nhược điểm của Javascript

**1.3.1 Ưu điểm**

JavaScript có rất nhiều ưu điểm khiến nó vượt trội hơn so với các đối thủ, đặc biệt trong các trường hợp thực tế. Sau đây chỉ là một số lợi ích của JavaScript:

* Bạn không cần một compiler vì web browser có thể biên dịch nó bằng HTML.
* Nó dễ học hơn các ngôn ngữ lập trình khác.
* Lỗi dễ phát hiện hơn và vì vậy dễ sửa hơn.
* Nó có thể được gắn trên một số element của trang web hoặc event của trang web như là thông qua click chuột hoặc di chuột tới.
* JS hoạt động trên nhiều trình duyệt, nền tảng, vân vân.
* Bạn có thể sử dụng JavaScript để kiểm tra input và giảm thiểu việc kiểm tra thủ công khi truy xuất qua database.
* Nó giúp website tương tác tốt hơn với khách truy cập.
* Nó nhanh hơn và nhẹ hơn các ngôn ngữ lập trình khác.

**1.3.2 Nhược điểm**

Mọi ngôn ngữ lập trình đều có các khuyết điểm. Một phần là vì ngôn ngữ đó khi phát triển đến một mức độ như JavaScript, nó cũng sẽ thu hút lượng lớn hacker, scammer, và những người có ác tâm luôn tìm kiếm những lỗ hổng và các lỗi bảo mật để lợi dụng nó. Một số khuyết điểm có thể kể đến là:

* Dễ bị khai thác.
* Có thể được dùng để thực thi mã độc trên máy tính của người dùng.
* Nhiều khi không được hỗ trợ trên mọi trình duyệt.
* JS code snippets lớn.
* Có thể bị triển khai khác nhau tùy từng thiết bị dẫn đến việc không đồng nhất.
* Dễ để học nhưng để trở thành master thì sẽ tốn khá nhiều thời gian

## Tìm hiểu về Framework ReactJs

### Tổng quan về ReactJs

ReactJs là một thư viện JavaScript dùng để xây dựng UI, UI ở đây được dùng chính ở 2 nền tảng Web và Mobile. Ở lĩnh vực Web, sử dụng React Js có thể đem lại trải nghiệm tốt cho người dùng, cũng như khả năng Hot Reload giúp bạn lập trình nhanh hơn. Còn ở lĩnh vực Mobile thì có React Native có thế sử dụng đa nền tảng cả trên ios và android.

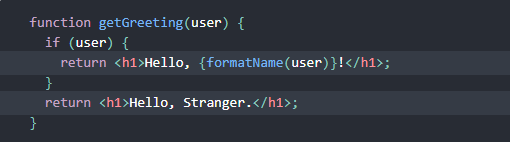
### Các khái niệm chính của ReactJs

1. *JSX*

JSX là một cú pháp mở rộng cho javascript, JSX gần giống với HTML cho ra những element (phần tử) React. JSX có thể kết hợp logic với giao diện, các sự kiện được xử lí như thế nào, state thay đổi ra sao.



Sau khi biên dịch (compile), biểu thức JSX là những gọi hàm bình thường của JavaScript và thành những đối tượng JavaScript sau khi được gọi. Có thể dùng JSX trong câu lệnh if else cũng như vòng lặp for loop. Gán nó cho biến, dùng như tham số hàm và trả về JSX từ hàm. JSX là một object.



1. *Rendering Elements*

Tạo một file html



Đây là nút DOM gốc vì mọi thứ trong nó sẽ được ReactDOM quản lý, các ứng dụng React thường có một nút DOM gốc.

Để kết xuất phần tử React thành nút DOM gốc thì chúng ta sử dụng ReactDOM.render():

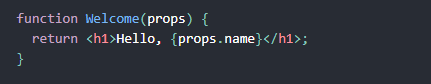


1. *Component và Props*

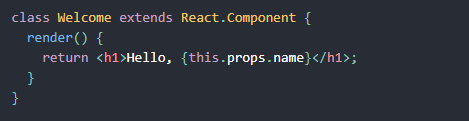
Component có thể chia UI thành các thành phần độc lập, có thể sử dụng riêng lẻ và tái sử dụng.

Về mặt khái niệm Component giống như các hàm javascript. Chúng nhận đầu vào tùy ý(được gọi là Props) và trả về các phần tử React mô tả những gì sẽ xuất hiện trên màn hình.

Đối với Function thì Props sẽ được thể hiện:



Còn đối với Class thì Props sẽ được biểu thị:

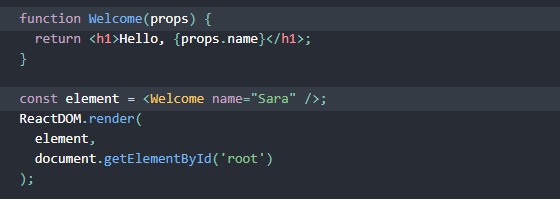


Có thể định nghĩa một Component thành một element.



Khi React thấy một element đại diện cho một thành phần do người dùng định nghĩa, nó sẽ chuyển thuộc tính JSX và các phần tử con cho các thành phần này dưới dạng một đối tượng. Và đó được gọi là Props.

Ví dụ:



1. *State và Lifecycle*

**State** là một đối tượng javascript lưu trữ dữ liệu động của một Component. State là dữ liệu động, nó cho phép một component theo dõi thông tin thay đổi ở giữa các kết xuất (render) và làm cho nó có thể tương tác. State trong React là bất biến, nghĩa là state không bao giờ nên được thay đổi trực tiếp. Và chỉ thay đổi được khi dùng setState. State thì được sử dụng trong một component.

Một khái niệm mới là global state xuất hiện khi có thêm thư viện Redux(giúp quản lý các state). Global state giúp chia sẻ state cho nhiều component sử dụng.

**Lifecycle** là một vòng đời của app. Các trạng thái của lifycycle là:

* Initialzation

Thời điểm khởi tạo Component, nó nằm trong constructor component. Tại đây, component có thể nhận các props truyền vào và thiết lập các giá trị cho State.

* Mounting

Ngay sau khi component được khởi tạo, hàm componentWillMount() được chạy và tại đây chúng ta có thể thực hiện những thao tác trước khi component được render(). Khi đã render được DOM rồi, chúng ta có thể thực hiện các thao tác update, kết nối với DB,... trong hàm componentDidMount().

* Updating

Hàm shouldComponentUpdate() kiểm tra xem states có được chúng ta thay đổi hay không, nếu có thay đổi nó sẽ chạy tiếp đến hàm componentWillUpdate(). Cũng giống với Mounting, hàm componentWillUpdate() được chạy ngay trước khi render và sau khi render xong rồi sẽ tiếp tục chạy đến hàm componentDidUpdate().

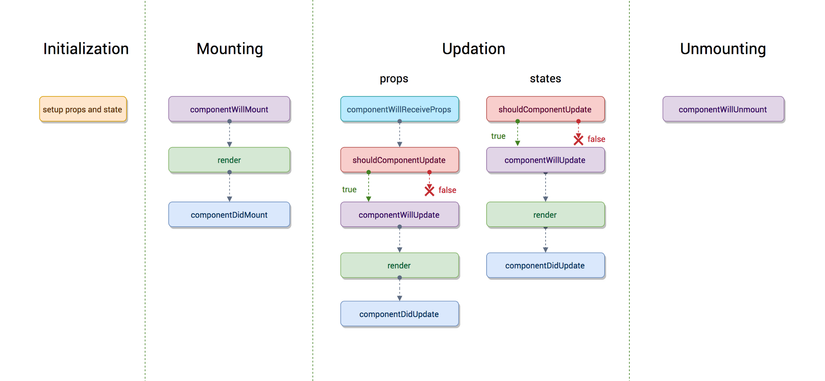
Tương tự cách hoạt động với states, khi nhận được sự thay đổi props truyền vào từ component cha, hàm componentWillReceiveProps() được thực hiện. Tại đây, component con được cập nhật lại giá trị cho từng props. Sau đó, trình tự thực hiện các hàm cho việc update props được thực hiện hoàn toàn tương tự với update states.

* Unmounting

Chúng ta chỉ có thể xóa một component mà không có ràng buộc với một component cha nào khác, tức là nó phải là component lớn nhất bao bọc các component con. Dùng hàm ReactDOM.unmountComponentAtNode để unmount một component và nó sẽ biến mất trên giao diện trang web.

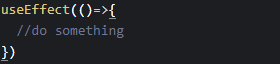
Lifecycle thì có các phương thức chính:

* ComponentWillMount()
* ComponentDidMount()
* ComponentDidUpdate()
* ComponentWillUnmounting()

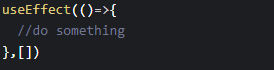


Nhưng ở hiện tại với ES6 ra đời và ReactHook thì người ta ưu tiên sử dụng ReactHook thay vì các phương thức cũ của lifecycle vì tính ngắn gọn, tiện lợi, dễ sử dụng của các Hook.

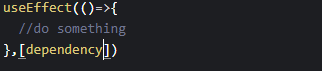
Ví dụ:



Sau mỗi lần render nó sẽ chạy lại và nó tương đương với ComponentWillMount()



Sẽ chạy một lần sau lần render đầu tiên và tương đương với phương thức ComponentDidMount()



Sẽ render lại mỗi khi dependency thay đổi và nó tương đương với phương thức ComponentDidUpdate()

1. *List và Keys*

## Tìm hiểu về thư viện Redux, Redux-Saga

## Tìm hiểu về PHP

# CHƯƠNG II: KHẢO SÁT HỆ THỐNG

# CHƯƠNG III: PHÂN TÍCH THIẾT KẾ HỆ THỐNG

# CHƯƠNG IV: DEMO WEBSITE