***React*** là một thư viện để xây dựng giao diện người dùng. ReactJS cho phép tạo ra các thành phần giao diện người dùng có thể tái sử dụng, hiển thị dữ liệu thay đổi theo thời gian. React trừu tượng hóa DOM, cung cấp một mô hình lập trình đơn giản hơn và hiệu suất tốt hơn. React cũng có thể hiển thị trên máy chủ bằng Node và phát triển ứng dụng mobile sử dụng react native. React triển khai luồng dữ liệu một chiều.

***Tính năng của React***

JSX là phần mở rộng cú pháp JavaScript.

Component – Làm việc với React là làm việc với các Component. Cần tư duy mọi thứ theo hướng component. Component giúp dễ dàng chỉnh sửa, phát triển khi làm việc trên các dự án quy mô lớn.

One-way và Flux – React triển khai luồng dữ liệu một chiều giúp dễ dàng suy luận về ứng dụng của mình. Flux là một mẫu giúp giữ dữ liệu theo một phương hướng duy nhất

***Ưu điểm của React***

Có thể sử dụng ReactJS để viết ứng dụng trực tiếp bằng JavaScript.

Biến các thành phần UI phức tạp trở thành những component độc lập.

Chuyển các dữ liệu đã được tùy biến đến một UI component cụ thể.

Có thể thay đổi trạng thái cho nhiều component con và không gây ảnh hưởng đến component gốc.

Xác định chính xác khi nào cần render lại cũng như khi nào bỏ đi phần tử DOM.

***Nhược điểm của React***

Reactjs chỉ phục vụ cho tầng View. React chỉ là View Library nó không phải là một MVC framework như những framework khác. React cũng không có 2-way binding hay là Ajax

Tích hợp Reactjs vào các framework MVC truyền thống yêu cầu cần phải cấu hình lại.

React khá nặng nếu so với các framework khác React có kích thước tương tương với Angular. Trong khi Angular là một framework hoàn chỉnh

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

***Toolchain*** là chuỗi công cụ tích hợp trong quá trình phát triển giúp đem đến trải nghiệm tốt nhất cho người dùng như:

* Mở rộng quy mô cho nhiều file và component.
* Sử dụng thư viện của bên thứ ba từ npm.
* Phát hiện sớm các lỗi thường gặp.
* Chỉnh sửa trực tiếp CSS và JS đang được phát triển.
* Tối ưu hóa đầu ra cho production

***Một số toolchain phổ biến***

NPM

[npm](https://www.npmjs.com/" \t "_blank) (Node Package Manager) là một công cụ quản lý gói cho Node.js và JavaScript. Nó cho phép người dùng tìm kiếm, cài đặt và quản lý các thư viện JavaScript được chia sẻ công khai trên npm Registry (cơ sở dữ liệu của npm)

Khi một dự án cần đến một thư viện từ npm, có thể sử dụng câu lệnh cài đặt thư viện đó thông qua CLI (Command line interface ), là giao diện dòng lệnh và là cách mà hầu hết các lập trình viên sử dụng để tương tác với npm.

Ví dụ về một câu lệnh cài đặt thư viện Next.js

npm install next

NPX

npx (Node Package Execute), là một công cụ để chạy package mà không cần phải cài đặt.

Để tạo một dự án Reactjs, sử dụng câu lệnh

npx create-react-app app-name

Với npx, không cần cài đặt package create-react-app vào dự án, package chỉ cần thực thi một lần với nhiệm vụ khởi tạo một dự án mới với các mô-đun cần thiết.

Yarn

Yarn là công cụ quản lý package tương tự như npm. Có thể thực hiện các câu lệnh để cài đặt, thực thi các câu lệnh thông qua CLI.

Webpack

webpack cho phép viết code mô-đun và gói nó lại với nhau thành các gói nhỏ để tối ưu hóa thời gian tải.

Babel

Babel là một trình biên dịch javascript, được sử dụng để chuyển đổi mã JavaScript chuẩn mới (ECMAScript 2015+) về phiên bản cũ hơn để đảm bảo chương trình có thể hoạt động bình thường trong các trình duyệt cũ.

Next.js

Next.js là một framework phổ biến và nhẹ cho ứng dụng static và server‑rendered được xây dựng bằng React.

Gatsby

Gatsby giúp tạo các website tĩnh với React. Nó cho phép sử dụng các component của React, nhưng xuất ra HTML và CSS được render trước để đảm bảo thời gian tải nhanh nhất.

***SSR - CSR***

Server side render (SSR). Bạn request một trang web, server xử lý nội dung thành HTML, return lại cho browser hiển thị lại lên màn hình.

Nội dung HTML được xử lý trên server và trả lại cho browser, lúc này nội dung HTML đã hoàn chỉnh và được hiển thị ngay khi nó được load về máy, tuy nhiên lúc này trang web mới chỉ xem được nội dung, người dùng chưa thể tương tác được. Ngay trong lúc này, trình duyệt vẫn âm thầm tải tiếp JS và thực thi nó ngay khi hoàn tất, một khi hoàn tất quá trình này người dùng mới có thể tương tác được. Từ đây có thể nhận thấy một vài điểm yếu nổi bật:

* Trang web phải xử lý lại hoàn toàn và load lại từ đầu nếu chỉ có một thay đổi nhỏ trong nội dung. (Ví dụ tiêu đề thay đổi …)
* Việc xử lý nội dung HTML khiến hao tốn tài nguyên server, gây chậm trễ khi xử lý các request khác.
* Lượng request lên server rất nhiều, do mọi tác vụ đều phải xử lý lại trên server và render lại HTML
* TTFB (Time To First Byte) cao do cần phải xử lý trên server.

Và từ những nhược điểm đó, Client Side Render (CSR) được sinh ra.

trình duyệt sẽ load tất cả nội dung về máy, một khi đã load thành công, nó sẽ thực thi code, lúc này nội dung trang web mới được hiển thị ra. Việc sinh nội dung … đều được thực hiện ngay trên client, server chỉ có nhiệm vụ trả mã nguồn JS và dữ liệu thô cho client. Với CSR, những nhược điểm đã nêu trên của SSR đã được giải quyết hết. Tuy nhiên, CSR không thể thay thế hoàn toàn được SSR. Nhược điểm của CSR nổi bật như :

* Lượng dữ liệu lần đầu load về khá nặng
* Nội dung hiển thị ra lần đầu chậm hơn so với SSR, tuy nhiên những lần sau thì quá nhanh do việc xử lý ngay trên client.
* SEO bị ảnh hưởng, do nội dung web được sinh trên client, khiến crawler của của Search engine không tiếp xúc được nội dung. Gần đây mới có Google giải quyết được vấn đề này