**1.React**

* **React là gì?**

ReactJS là một thư viện JavaScript mã nguồn mở phát triển bởi Facebook và cộng đồng các nhà phát triển trên toàn thế giới. Với đặc tính linh hoạt, đơn giản và dễ sử dụng, ReactJS đã trở thành một trong những công nghệ phát triển web được ưa chuộng nhất hiện nay. Trong bài viết này, Stringee sẽ cùng bạn tìm hiểu về ReactJS - từ lịch sử phát triển, tính năng, cách sử dụng và các dự án thành công đã sử dụng ReactJS để xây dựng [1].

* **Các tính năng nổi bật của ReactJS**

ReactJS có nhiều tính năng hữu ích cho việc phát triển ứng dụng web, bao gồm:

* **Components**: ReactJS cho phép phát triển ứng dụng web theo mô hình component. Các component là các phần tử UI độc lập có thể được tái sử dụng trong nhiều phần khác nhau của ứng dụng.
* **Virtual DOM**: ReactJS sử dụng Virtual DOM để tối ưu hóa hiệu suất của ứng dụng. Virtual DOM là một bản sao của DOM được lưu trữ trong bộ nhớ và được cập nhật một cách nhanh chóng khi có thay đổi, giúp tăng tốc độ và hiệu suất của ứng dụng.
* **JSX**: JSX là một ngôn ngữ lập trình phân biệt được sử dụng trong ReactJS để mô tả các thành phần UI. JSX kết hợp HTML và JavaScript, giúp cho việc viết mã dễ hiểu và dễ bảo trì hơn.
* **State và Props**: ReactJS cho phép quản lý trạng thái của các thành phần UI thông qua State và Props. State là trạng thái của một thành phần được quản lý bởi nó chính, trong khi Props là các giá trị được truyền vào từ bên ngoài để tùy chỉnh hoặc điều khiển hành vi của một thành phần.
* **Hỗ trợ tốt cho SEO**: ReactJS hỗ trợ tốt cho việc tối ưu hóa SEO. Với các thư viện như React Helmet, các nhà phát triển có thể quản lý các phần tử meta và title cho từng trang web, giúp tăng khả năng tìm kiếm và tăng cường trải nghiệm người dùng.
* **Hỗ trợ đa nền tảng**: ReactJS không chỉ được sử dụng để phát triển ứng dụng web, mà còn được sử dụng để phát triển ứng dụng di động với React Native. Sử dụng React Native, các nhà phát triển có thể xây dựng ứng dụng di động cho cả iOS và Android sử dụng cùng một mã nguồn.
* **Redux**: Redux là một thư viện quản lý trạng thái cho các ứng dụng ReactJS. Nó giúp quản lý trạng thái của ứng dụng một cách chính xác và dễ dàng, đồng thời giúp tăng tính linh hoạt và khả năng mở rộng của ứng dụng.
* **Cách sử dụng ReactJS trong phát triển web**

Để sử dụng ReactJS trong phát triển web, bạn có thể làm theo các bước sau:

* Bước 1 - Cài đặt Node.js và npm: ReactJS được xây dựng trên nền tảng Node.js, do đó bạn cần cài đặt Node.js và npm để phát triển ứng dụng ReactJS.
* Bước 2 - Tạo một ứng dụng React: Bạn có thể tạo một ứng dụng React bằng cách sử dụng lệnh "create-react-app" trong Command Prompt hoặc Terminal.
* Bước 3 - Tạo các component: Tạo các component để xây dựng giao diện người dùng cho ứng dụng của bạn. Bạn có thể tạo component bằng cách sử dụng class hoặc hàm.
* Bước 4 - Xây dựng giao diện người dùng: Sử dụng JSX để xây dựng giao diện người dùng cho ứng dụng của bạn. JSX là một ngôn ngữ phân biệt được sử dụng trong ReactJS để mô tả các thành phần UI.
* Bước 5 - Quản lý trạng thái: Sử dụng State và Props để quản lý trạng thái của các thành phần UI. State là trạng thái của một thành phần được quản lý bởi nó chính, trong khi Props là các giá trị được truyền vào từ bên ngoài để tùy chỉnh hoặc điều khiển hành vi của một thành phần.
* Bước 6 - Kết nối với API: Sử dụng thư viện như Axios để kết nối với API và lấy dữ liệu từ server.
* Bước 7 - Build và triển khai ứng dụng: Sử dụng lệnh "npm run build" để build ứng dụng của bạn và triển khai nó trên môi trường sản phẩm.
* Tóm lại, để sử dụng ReactJS trong phát triển web, bạn cần cài đặt Node.js và npm, tạo các component, xây dựng giao diện người dùng, quản lý trạng thái, kết nối với API, build và triển khai ứng dụng.

**2. NodeJS**

* **NodeJS là gì?**

Node.js là một nền tảng mã nguồn mở được xây dựng trên V8 Javascript engine. Node.js được viết bằng c++ và Javascript. Nền tảng này được phát triển Node.js vào năm 2009 bởi Ryan Lienhart Dahl.

Mục đích của Node.js là cung cấp một môi trường chạy ứng dụng web hiệu quả và không đồng bộ. Nền tảng này sử dụng mô hình I/O không chặn, cho phép xử lý nhiều yêu cầu cùng lúc mà không bị tắc nghẽn [2].

* **Các thành phần của Node.JS**
* **Module**

Module trong Node.js giống như những gói thư viện nhỏ, chứa các hàm, đối tượng và các lớp được viết sẵn để thực hiện các tác vụ cụ thể. Điều này giúp cho việc phát triển ứng dụng trở nên modun hóa và dễ dàng quản lý. Để sử dụng một module trong ứng dụng của bạn, bạn chỉ cần sử dụng hàm require(). Ví dụ:

const http = require('http');

Ở Node.JS có nhiều Module với nhiều chức năng khác nhau và đều cần thiết cho một ứng dụng web. Dưới đây là một số Module thường được sử dụng trong ứng dụng web:

|  |  |
| --- | --- |
| Module chính | Mô tả |
| http | Tạo và quản lý các máy chủ HTTP để xây dựng các ứng dụng web |
| util | Cung cấp các hàm tiện ích hỗ trợ lập trình, như định dạng dữ liệu, kiểm tra kiểu dữ liệu… |
| fs | Tương tác với hệ thống file để đọc, ghi, xóa và quản lý các file |
| url | Phân tích cú pháp URL để trích xuất các thông tin như protocol, host, path… |
| querystring | Xử lý query string trong URL để lấy ra các tham số truyền vào |
| stream | Làm việc với các luồng dữ liệu một cách hiệu quả, giúp tối ưu hóa việc xử lý dữ liệu lớn |
| zlib | Nén và giải nén dữ liệu để giảm kích thước file và tăng tốc độ truyền tải |

* **Bảng điều khiển**

Bảng điều khiển là một module có chức năng hỗ trợ việc gỡ lỗi các ứng dụng JavaScript. Bảng điều khiển này cung cấp một giao diện tương tác để in ra các thông tin quan trọng như giá trị của biến, kết quả của biểu thức và các thông báo lỗi.

Một trong những phương pháp phổ biến để sử dụng bảng điều khiển là sử dụng hàm console.log(). Hàm này cho phép in ra bất kỳ giá trị nào lên màn hình console. Ví dụ:

console.log('Hello, world!');

* **Cluster**

NodeJS, nổi tiếng với khả năng lập trình không đồng bộ, thường được xây dựng dựa trên mô hình đơn luồng. Tuy nhiên, để cải thiện hiệu năng và tận dụng tối đa tài nguyên CPU, Cluster là một giải pháp hoàn hảo.

Node.JS có khả năng lập trình không đồng bộ nên thường được xây dựng trên mô hình đơn luồng

Module Cluster trong NodeJS cho phép bạn tạo ra nhiều tiến trình con (worker processes), mỗi tiến trình sẽ xử lý các yêu cầu đến. Điều này giúp cân bằng tải và ngăn chặn việc một tiến trình bị quá tải. Các worker process này chia sẻ cùng một cổng máy chủ, tạo nên một cụm các tiến trình làm việc cùng nhau.

* **Đối tượng toàn cục**

Trong Node.JS các đối tượng toàn cục đóng vai trò vô cùng quan trọng. Đây là những biến hoặc hàm được sẵn có trong mọi module của NodeJS, cho phép các lập trình viên truy cập và sử dụng chúng trực tiếp mà không cần khai báo lại. Một số đối tượng toàn cục phổ biến ở trong bảng dưới đây:

|  |  |
| --- | --- |
| Đối tượng toàn cục | Mô tả |
| \_\_\_dirname | Trả về đường dẫn tuyệt đối đến thư mục chứa file đang thực thi. |
| \_\_\_filename | Trả về đường dẫn tuyệt đối đến file đang thực thi. |
| exports | Dùng để xuất các biến hoặc hàm từ module hiện tại. |
| module | Biểu diễn module hiện tại. |
| require | Dùng để nhập các module khác vào module hiện tại. |

* **Xử lý lỗi**

Trong ứng dụng của Node.JS thường sẽ gặp 4 lỗi phổ biến như sau:

|  |  |
| --- | --- |
| Lỗi JavaScript | Lỗi này xảy ra do cú pháp JavaScript không hợp lệ hoặc lỗi thời gian chạy. Ví dụ bao gồm EvalError, SyntaxError, RangeError, ReferenceError, TypeError và URIError. |
| Lỗi từ hệ thống | Lỗi này phát sinh từ các vấn đề cấp hệ thống, chẳng hạn như tệp không tồn tại hoặc đóng socket. |
| Lỗi từ người dùng | Lỗi này do logic hoặc đầu vào được xác định bởi người dùng gây ra. |
| Lỗi assertion | Lỗi này cho thấy vi phạm các điều kiện hoặc logic mong đợi. |

* **Streaming (Luồng)**

- Luồng (Streaming) là một dòng dữ liệu liên tục được truyền từ nguồn đến đích. Trong lập trình, luồng được sử dụng để đọc hoặc ghi dữ liệu vào các thiết bị ngoại vi như file, mạng, hoặc các bộ nhớ đệm (buffer).

- Luồng là một dòng dữ liệu được truyền liên tục từ nguồn đến đích

- Có 4 loại luồng chính:

+ Luồng đọc: Chỉ cho phép đọc dữ liệu từ nguồn.

+ Luồng ghi: Chỉ cho phép ghi dữ liệu vào đích.

+ Luồng duplex: Cho phép vừa đọc vừa ghi dữ liệu.

+ Luồng chuyển đổi: Thay đổi định dạng hoặc mã hóa dữ liệu trong quá trình truyền.

Ví dụ: Khi bạn xem một video trực tuyến, dữ liệu video được truyền dưới dạng một luồng liên tục từ máy chủ đến trình duyệt của bạn.

* **Buffer (Bộ nhớ đệm)**

Buffer là một cấu trúc dữ liệu được sử dụng để lưu trữ và thao tác hiệu quả với dữ liệu nhị phân. Buffer cung cấp một cách linh hoạt để làm việc với dữ liệu thô trong bộ nhớ.

* **Tên miền (Domain)**

Module tên miền giúp ngăn chặn và khắc phục các lỗi tiềm ẩn trong website của bạn. Module này sử dụng hai cơ chế chính để đảm bảo hoạt động ổn định của hệ thống:

* **Liên kết nội bộ:** Bộ phát lỗi sẽ được tích hợp trực tiếp vào mã nguồn của website, giúp xác định và xử lý lỗi một cách nhanh chóng và hiệu quả.
* **Liên kết bên ngoài:** Bạn có thể thêm thủ công các bộ phát lỗi vào tên miền thông qua cấu hình hệ thống. Điều này đặc biệt hữu ích khi bạn muốn theo dõi và quản lý các lỗi từ các nguồn bên ngoài.
* **DNS**

Module DNS dùng để tương tác với các máy chủ Hệ thống tên miền (DNS). Module DNS cho phép các nhà phát triển hiệu quả phân giải tên miền thành địa chỉ IP.

- Có hai phương pháp chính để phân giải DNS:

dns.resolve(): Phương pháp này thiết lập kết nối mạng với máy chủ DNS để thực hiện quá trình phân giải. Phù hợp cho các ứng dụng yêu cầu tra cứu DNS thời gian thực hoặc cần xử lý các truy vấn DNS phức tạp.

dns.lookup(): Phương pháp này thực hiện phân giải DNS mà không cần kết nối mạng. Phù hợp cho các trường hợp kết nối mạng bị hạn chế hoặc không đáng tin cậy, chẳng hạn như ứng dụng ngoại tuyến hoặc cơ chế lưu trữ cache.

* **Debugger (Trình gỡ lỗi)**

Node.JS có công cụ gỡ lỗi tích hợp vô cùng tiện lợi, giúp các nhà phát triển kiểm tra và khắc phục lỗi trong mã nguồn một cách hiệu quả. Mặc dù không sở hữu nhiều tính năng phức tạp như các trình gỡ lỗi chuyên dụng, nhưng công cụ này hoàn toàn đáp ứng nhu cầu kiểm tra code đơn giản trong quá trình phát triển ứng dụng.

Để bắt đầu sử dụng công cụ gỡ lỗi, bạn chỉ cần mở terminal và sử dụng lệnh inspect trước tên tệp JavaScript mà bạn muốn kiểm tra. Ví dụ:

$ node inspect myscript.js

* **Một số thuật ngữ liên quan đến Node.JS**

Để hiểu rõ hơn về Node.JS, chúng ta cần nắm bắt một số thuật ngữ cơ bản. Trong phần tiếp theo này, chúng ta sẽ tìm hiểu về các khái niệm như I/O, không đồng bộ, không chặn và sự kiện và lập trình hướng sự kiện.

* **I/O**

I/O là từ viết tắt của input/output là thuật ngữ chủ yếu nói về sự tương tác không đồng bộ của chương trình với hệ thống.

Ví dụ: Các hoạt động của I/O có thể bao gồm việc đọc/ghi dữ liệu từ/vào disk, tạo các yêu cầu HTTP và trao đổi với cơ sở dữ liệu. Hoạt động này rất chậm so với việc truy cập bộ nhớ (RAM) hoặc thực hiện công việc trực tiếp trên CPU.

Để tối ưu hiệu năng I/O trong NodeJS, các lập trình viên thường sử dụng các kỹ thuật như non-blocking I/O, asynchronous programming và event-driven architecture.

* **Không đồng bộ**

Không đồng bộ (Asynchronous) nói về việc chạy không theo trình tự xuất hiện bên trong code. Trong lập trình không đồng bộ, một chương trình có thể không chờ cho một tác vụ được hoàn thành mà có thể chuyển ngay sang một tiếp vụ tiếp theo.

Asynchronous nghĩa là việc chạy không theo trình tự xuất hiện bên trong code

* **Không chặn**

Chặn nghĩa là một tác vụ sẽ tạm dừng việc thực thi của chương trình cho đến khi chương trình đó hoàn tất. Ví dụ, khi bạn gọi một hàm đọc file, chương trình sẽ đợi cho đến khi file được đọc xong mới tiếp tục thực hiện các lệnh tiếp theo. Điều này có thể gây ra tình trạng “đứng hình” nếu tác vụ đó mất nhiều thời gian.

Trong Node.JS chương trình được phép tiếp tục thực hiện một tác vụ khác trong khi chờ kết quả từ một tác vụ

Ngược lại, không chặn cho phép chương trình tiếp tục thực hiện các công việc khác trong khi một tác vụ đang chờ kết quả. Các phương thức không đồng bộ này đặc biệt hữu ích trong các LLM vì chúng giúp tăng hiệu suất và khả năng xử lý nhiều yêu cầu cùng một lúc.

* **Sự kiện và Lập trình Hướng sự kiện**

Sự kiện là những tương tác do người dùng hay hệ thống tạo nên, có thể là một cú click chuột, một lần kéo thả hoặc thậm chí là một thay đổi trạng thái của hệ thống. Mỗi sự kiện này đều mang thông tin quan trọng, giúp ứng dụng hiểu được yêu cầu của người dùng và phản hồi một cách thích hợp.

Lập trình hướng sự kiện là một mô hình lập trình linh hoạt, cho phép chương trình phản ứng nhanh chóng trước các tương tác của người dùng. Thay vì thực hiện theo một trình tự cố định, chương trình sẽ chờ các sự kiện xảy ra và gọi đến các hàm tương ứng để xử lý.

* **Các tính chất của Node.JS**
* **Không đồng bộ về bản chất và Hướng sự kiện**

Với mô hình không đồng bộ và hướng sự kiện, các máy chủ NodeJS hoạt động vô cùng hiệu quả. Thay vì chờ đợi dữ liệu từ mỗi API, NodeJS sẽ chuyển trực tiếp sang các tác vụ khác, tận dụng tối đa tài nguyên hệ thống.

Nhờ cơ chế hướng sự kiện, NodeJS có thể nhận và theo dõi tất cả các phản hồi API một cách linh hoạt. Điều này giúp các ứng dụng NodeJS trở nên non-blocking, đảm bảo hiệu suất cao và khả năng xử lý lượng lớn yêu cầu đồng thời.

* **Kiến trúc đơn luồng**

Node.js hoạt động dựa trên một luồng chính duy nhất, nhưng vẫn có thể xử lý đồng thời nhiều yêu cầu từ máy khách. Điều này là nhờ vào mô hình vòng lặp sự kiện.

Node.JS dựa trên một luồng chính duy nhất

Vòng lặp sự kiện cho phép Node.js không bị chặn bởi các hoạt động I/O như đọc ghi file hay truy vấn cơ sở dữ liệu. Nhờ đó, Node.js trở thành lựa chọn hàng đầu cho các ứng dụng thời gian thực và I/O-bound.

[1] <https://stringee.com/vi/blog/post/reactJS-la-gi>

[2] <https://vietnix.vn/nodejs-la-gi/>