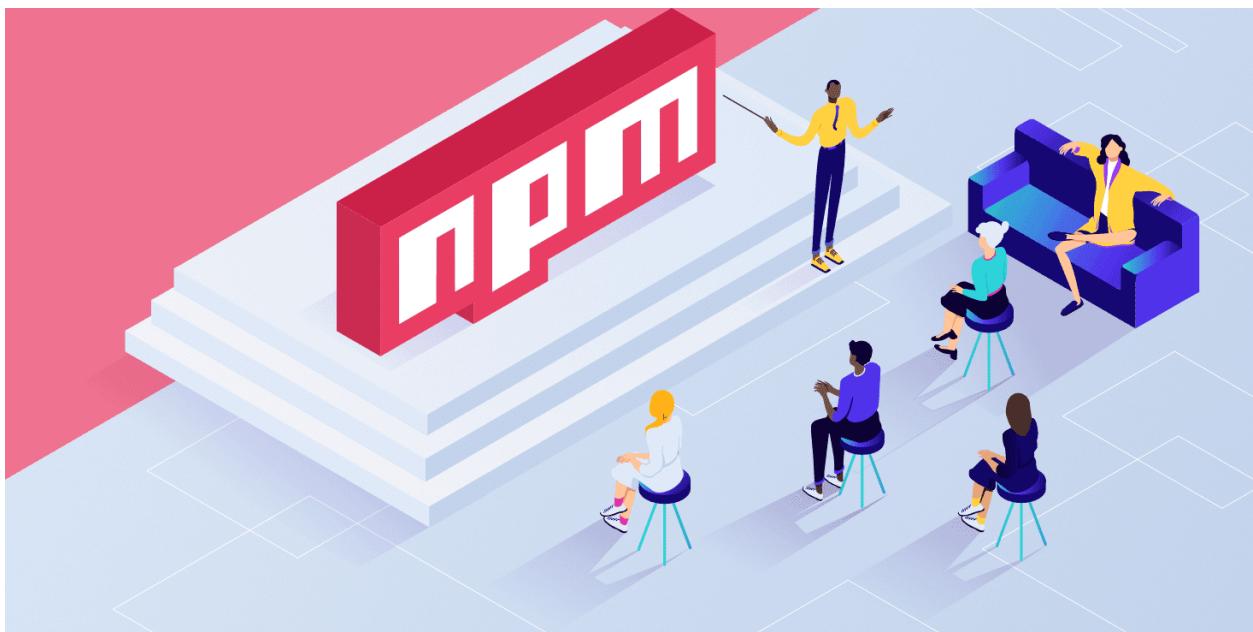


1. **HTML (HyperText Markup Language)**: Ngôn ngữ đánh dấu siêu văn bản được sử dụng để tạo cấu trúc cho trang web.
2. **CSS (Cascading Style Sheets)**: Dùng để miêu tả cách thức hiển thị của các thành phần HTML trên trang web, bao gồm màu sắc, phông chữ, bố cục và nhiều thuộc tính khác...
3. **JavaScript (JS)**: Ngôn ngữ lập trình chính được sử dụng để tạo ra các trang web có thể tương tác, xử lý logic...
4. **DOM (Document Object Model)**: Đây là một giao diện lập trình ứng dụng (API) được sử dụng để truy cập và thao tác trên các tài liệu dạng HTML và XML. DOM thường được biểu diễn dưới dạng một cây cấu trúc dữ liệu, giúp mô tả tài liệu một cách trực quan và dễ hiểu.
5. **Framework**: Bộ công cụ hoặc thư viện giúp việc phát triển front-end trở nên dễ dàng và hiệu quả hơn. Các framework phổ biến bao gồm:
 - o **React**: Một thư viện JavaScript để xây dựng giao diện người dùng.
 - o **Angular**: Một framework phát triển bởi Google, dùng để xây dựng ứng dụng web động.
 - o **Vue.js**: Một framework JavaScript linh hoạt và nhẹ nhàng để xây dựng giao diện người dùng.
6. **Bootstrap**: Một framework front-end phổ biến giúp tạo ra các trang web responsive nhanh chóng với các thành phần CSS sẵn có.
7. **SASS (Syntactically Awesome Style Sheets)**: Một tiền xử lý CSS giúp viết mã CSS một cách dễ dàng và hiệu quả hơn bằng cách sử dụng biến, lồng, và các quy tắc kế thừa (ngoài ra còn có SCSS, LESS) mà bạn có thể tham khảo thêm.
8. **Webpack**: Một module bundler cho JavaScript, cho phép nén và tối ưu hóa mã nguồn của bạn.

9. **NPM (Node Package Manager)**: Một công cụ quản lý gói cho JavaScript, thường được sử dụng để cài đặt và quản lý các thư viện và công cụ front-end.



10. **Version Control (Git)**: Hệ thống quản lý phiên bản, giúp theo dõi sự thay đổi trong mã nguồn và hỗ trợ làm việc nhóm hiệu quả.

11. **API (Application Programming Interface)**: Giao diện lập trình ứng dụng, giúp front-end giao tiếp với back-end hoặc các dịch vụ bên ngoài.

12. **AJAX (Asynchronous JavaScript and XML)**: Kỹ thuật sử dụng JavaScript để gửi và nhận dữ liệu từ máy chủ mà không cần tải lại toàn bộ trang.

13. **SEO (Search Engine Optimization)**: Tối ưu hóa công cụ tìm kiếm, giúp trang web của bạn có thể dễ dàng được tìm thấy trên các công cụ tìm kiếm như Google.

14. **BEM (Block Element Modifier)**: Một phương pháp để viết CSS theo một cấu trúc nhất định và dễ bảo trì.

15. **ES6 (ECMAScript 2015)**: Phiên bản thứ sáu của tiêu chuẩn ECMAScript, cung cấp nhiều tính năng mới cho JavaScript như arrow functions, classes, modules, và promises...

16. **Polyfill**: Một đoạn mã (thường là JavaScript) được thêm vào trang web để cung cấp tính năng mà trình duyệt cũ không hỗ trợ.

17. **PWA (Progressive Web App)**: Ứng dụng web tiến bộ, cung cấp trải nghiệm người dùng tương tự như ứng dụng gốc với khả năng làm việc offline và thông báo đẩy.

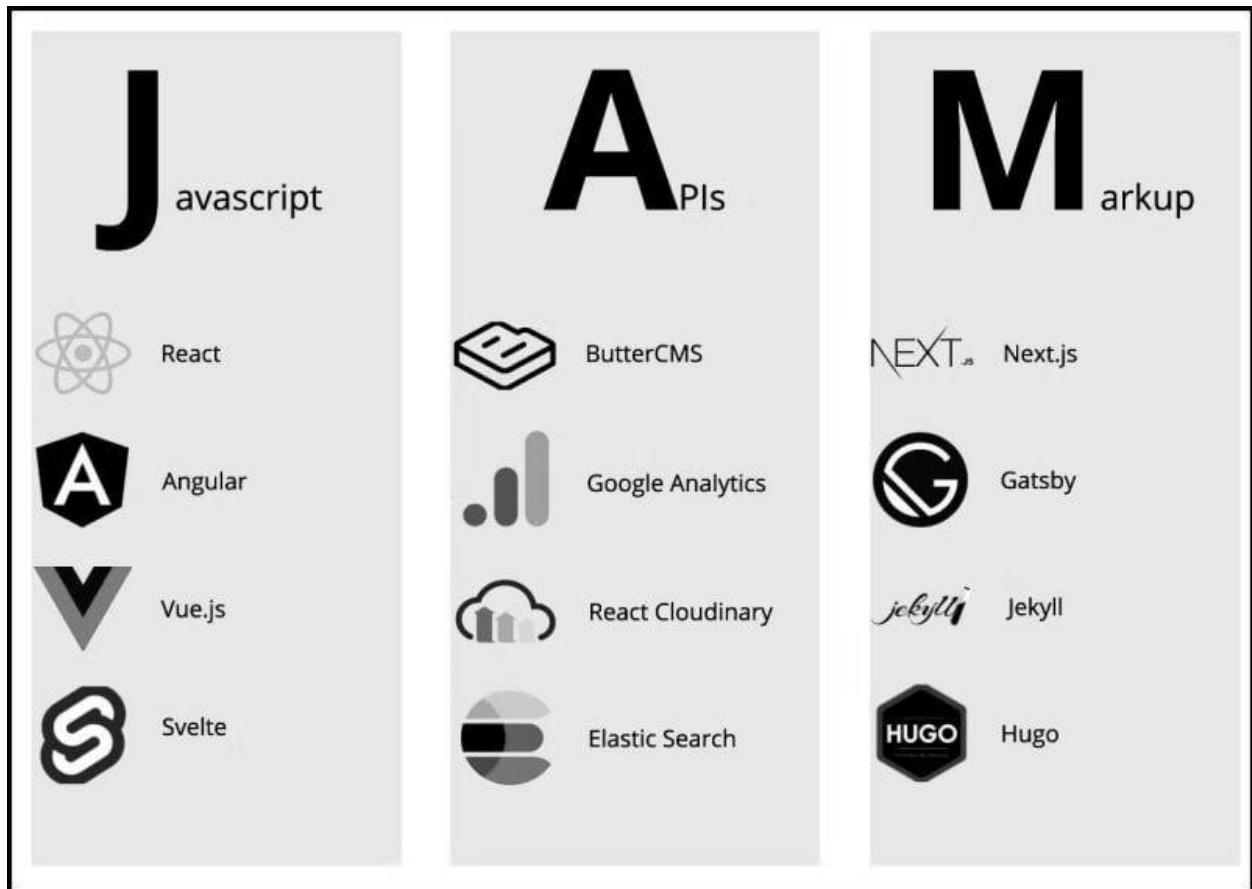
18. **Single Page Application (SPA)**: Ứng dụng trang đơn, nơi toàn bộ trang web được tải về chỉ một lần và các tương tác tiếp theo được xử lý bởi JavaScript mà không cần tải lại trang.
19. **SSR (Server-Side Rendering)**: Kỹ thuật render nội dung trang web trên máy chủ và gửi HTML hoàn chỉnh đến trình duyệt, cải thiện tốc độ tải trang và SEO.
20. **CSR (Client-Side Rendering)**: Kỹ thuật render nội dung trang web trên phía client (trình duyệt), thường được sử dụng trong các ứng dụng SPA.
21. **RESTful API**: Giao diện lập trình ứng dụng tuân theo nguyên tắc REST, sử dụng HTTP để giao tiếp giữa client và server.
22. **GraphQL**: Ngôn ngữ truy vấn cho API, cho phép client yêu cầu chính xác những gì họ cần.
23. **JSON (JavaScript Object Notation)**: Định dạng dữ liệu nhẹ, dễ đọc và ghi, được sử dụng phổ biến để trao đổi dữ liệu giữa server và client.



24. **JWT (JSON Web Token)**: Một tiêu chuẩn mở (RFC 7519) được sử dụng để truyền thông tin xác thực an toàn giữa các bên. JWT thường được sử dụng để xác thực người dùng trong ứng dụng web và di động.
25. **OAuth**: Giao thức ủy quyền mở, cho phép các ứng dụng truy cập vào tài nguyên người dùng trên một dịch vụ khác mà không cần cung cấp mật khẩu.

26. **Transpiling**: Quá trình chuyển đổi mã nguồn từ ngôn ngữ lập trình này sang ngôn ngữ lập trình khác (Ví dụ: chuyển đổi từ TypeScript sang JavaScript).
27. **CORS (Cross-Origin Resource Sharing)**: Cơ chế cho phép hoặc hạn chế các yêu cầu tài nguyên từ một domain khác với domain mà tài nguyên đó đang được phục vụ.
28. **Responsive Web Design (RWD)**: Thiết kế web đáp ứng, sử dụng các kỹ thuật như media queries để làm cho trang web hiển thị tốt trên nhiều thiết bị và kích thước màn hình khác nhau.
29. **Lazy Loading**: Kỹ thuật trì hoãn việc tải các tài nguyên (như hình ảnh hoặc video) cho đến khi chúng thực sự cần thiết để cải thiện tốc độ tải trang.
30. **Minification**: Quá trình loại bỏ các ký tự không cần thiết (như khoảng trắng, dấu xuống dòng) khỏi mã nguồn mà không ảnh hưởng đến chức năng của nó, nhằm giảm kích thước file và tăng tốc độ tải trang.
31. **Service Worker**: Một script chạy trong nền trình duyệt, giúp ứng dụng web làm việc offline và xử lý các thông báo đẩy (push notification).
32. **LocalStorage**: Một phần của Web Storage API, cho phép lưu trữ dữ liệu phía client vĩnh viễn trong trình duyệt.
33. **SessionStorage**: Một phần của Web Storage API, cho phép lưu trữ dữ liệu phía client trong một phiên làm việc của trình duyệt.
34. **WebSocket**: Một giao thức giao tiếp hai chiều, giúp thiết lập kết nối lâu dài giữa client và server, thích hợp cho các ứng dụng real-time.
35. **Virtual DOM**: Một khái niệm trong các framework như React, là một bản sao của DOM được lưu trữ trong bộ nhớ để tối ưu hóa hiệu suất.
36. **Client-side Routing**: Quản lý việc điều hướng trang web trong các ứng dụng SPA mà không cần tải lại trang.
37. **Code Splitting**: Kỹ thuật chia nhỏ mã nguồn thành các phần nhỏ hơn để tải chỉ những phần cần thiết khi người dùng tương tác, giúp tăng tốc độ tải trang.
38. **Babel**: Một công cụ chuyển đổi mã (transpiler) cho JavaScript, thường được sử dụng để chuyển đổi ES6+ sang ES5 để tương thích với nhiều trình duyệt hơn.
39. **PostCSS**: Một công cụ xử lý CSS với các plugin có thể chuyển đổi CSS bằng JavaScript .

40. **ESLint**: Một công cụ linting cho JavaScript, giúp phát hiện và sửa các lỗi cú pháp và style mã hóa.
41. **Lighthouse**: Một công cụ tự động mã nguồn mở của Google để cải thiện chất lượng trang web, bao gồm hiệu suất, khả năng truy cập và SEO.
42. **Content Delivery Network (CDN)**: Một hệ thống các máy chủ phân phối nội dung nhanh chóng cho người dùng dựa trên vị trí địa lý của họ.
43. **Micro Frontends**: Một kiến trúc xây dựng ứng dụng web bằng cách phân tách thành các phần nhỏ hơn, mỗi phần có thể được phát triển và triển khai độc lập.
44. **Atomic Design**: Một phương pháp thiết kế giao diện người dùng bằng cách phân chia thành các thành phần nhỏ hơn như atoms, molecules, organisms, templates, và pages.
45. **JAMstack**: Một kiến trúc hiện đại dựa trên JavaScript, APIs và Markup, giúp tạo ra các trang web nhanh chóng và bảo mật.



46. **Headless CMS**: Hệ thống quản lý nội dung cung cấp API để tách biệt phần quản trị và phần hiển thị nội dung.

47. **Web Vitals**: Các chỉ số hiệu suất chính của trang web do Google đề xuất, bao gồm LCP (Largest Contentful Paint), FID (First Input Delay), và CLS (Cumulative Layout Shift).
48. **Sourcemaps**: Tập tin giúp gỡ lỗi JavaScript dễ dàng hơn bằng cách ánh xạ mã nguồn đã biên dịch trở lại mã nguồn gốc của nó.
49. **Flexbox**: Một mô hình bố cục CSS, cung cấp một cách dễ dàng và hiệu quả để bố trí, căn chỉnh và phân phối không gian giữa các mục trong một container.
50. **Grid Layout**: Một hệ thống bố cục CSS hai chiều, cho phép tạo ra các thiết kế web phức tạp và đáp ứng với các hàng và cột.
51. **Critical CSS**: một kỹ thuật **tối ưu hóa hiệu suất** cho website bằng cách **xác định và tải** chỉ những **CSS cần thiết** cho phần **trên màn hình đầu tiên (above-the-fold)** của trang web. Điều này giúp cải thiện **tốc độ tải trang** và **trải nghiệm người dùng**.
52. **Pre-rendering**: Kỹ thuật tạo sẵn HTML tĩnh cho các trang để cải thiện hiệu suất tải trang và SEO.
53. **Hydration**: là quá trình **chuyển đổi một trang web tĩnh (static website)** thành một trang web **động (dynamic website)** bằng cách **thêm các thành phần JavaScript tương tác**. Quá trình này thường được sử dụng với các framework **server-side rendering (SSR)** như **Next.js** và **Nuxt.js**.