NEXTJS

## 1, SSG, SSR, CSR, ISR

Graphical user interface, text

Description automatically generated

Graphical user interface

Description automatically generated

Mô tả quá trình đặc trưng khi viết react bên phía server

+, Viết code bằng react

+, Build ở mode production, sử dụng ReactDOMServer để build ra các file HTML

+, Khi user request lên server, trả file HTML đó về, file HTML đó đã được render sẵn từ layout, link,…. Tuy nhiên chỉ là phần content, chưa có phần event, xử lý Javascript.

+, Sau khi user log HTML lên, sau đó log tiếp file Javascript => gọi là quá trình hydration. Sử dụng hàm ReactDOM.hydrate() để attach các Event Listener cho HTML được render bên phía server

- Pre-rendering

Diagram

Description automatically generated with low confidence

Sẽ render sẵn file HTML ở phía server, khi user log lên thì dã có sẵn HTML để trả về show lên rồi. Nhưng chưa có gắn Event ngay từ lúc Initial. Sau đó sẽ load JS và thưc hiện quá trình Hydration.

- SSG – Static Site Generation

Diagram

Description automatically generated

Hiện tại SSG được khuyến khích và sử dụng default của NextJs.

Tại thời điểm build dự án, build lên đã ra được sẵn nội dung static. Mỗi khi user request lên thì chỉ cần trả về file static được tạo ra sẵn đó lúc build là xong.

Diễn ra trong quá trình build time

- SSR – Server Side Rendering

Diagram

Description automatically generated

Nếu như ở dạng each request: user gửi request lên, sau đó phải đi xử lý và gom dự liệu gì đó, tạo ra 1 file HTML và trả về cho user. Mỗi lần request lên đều phải làm các việc như trên => request nhiều thì server rất cực và tốn resource. Server tính toán nhanh hay chậm thì user cũng phải đợi nhanh hay chậm

- CSR – Client Side Rendering

- ISR – Incrêmntal Static Regeneration

## 2, Automatic Static Optimization

Graphical user interface, text, application, email

Description automatically generated

- Ý nghĩa: nextjs xác định 1 trang có phải static hay không hay có pre-rendered được hay không? Xác định bằng cách mình có sử dụng hàm **getServerSideProps** hay **getInitialProps**. Nếu có sử dụng thì trang đó sẽ bị rendered theo các request, không thể nào prerendered trước đó được.

=> Nếu sử dụng 2 hàm đó thì không phải ASO

- Nếu trang sử dụng ASO => khi gọi next build => output sau khi build ra là file HTML

- Nếu trang không sử dụng ASO => next build output là 1 file JS => phải load JS về và chạy file đó mới ra được muckup

- Khi sử dụng router.query, đối với ASO lần đầu tiên object query sẽ bị empty. Nhưng sau khi hydration xong, sẽ trigger 1 lần update nếu query đó có sự thay đổi, nếu không sẽ ko trigger update nữa. Còn đối với no-ASO điều này luôn available.

NOTE: Cần lưu ý trong trường hợp sử dụng useEffect để fetch dữ liệu bên phía client, cần 1 deps router.query. Cần check query có available hay không thì mới gọi api, nếu ko check thì lần đầu tiên object query sẽ bị rỗng => Sẽ bị lỗi.

useEffect(() => {

// check query available

}, [router.query])

## 3, Static Site Generation – SSG

Diagram

Description automatically generated with medium confidence

- Cách hoạt động: build ra những file HTML khi chạy dòng lệnh “yarn build” và build ra các file HTML sẵn. Server đã có sẵn các file HTML rồi, khi user request lên chỉ cần trả HTML về là xong không cần xử lý thêm gì cả => build ra HTML và user tái sử dụng là xong.

- SSG có nhiều loại

+, Static HTML Generation: chỉ có HTML thôi không có JSON data gì cả

+, Static HTML Generation + JSON Data khi sử dụng thêm hàm getStaticProps

+, Static HTML Generation + JSON Data + Dynamic Routes: sử dụng thêm hàm getStaticProps + getStaticPaths

- Khi nào sử dụng

+, Nếu page **content** phụ thuộc vào external data => sử dụng getStaticProps

+, Nếu page **paths** phụ thuộc vào external data => sử dụng getStaticPaths (ví dụ: có 10 bài blog, blog/blog1, blog/blog2,…)

- SSG và SSR không thể sử dụng chung với nhau

\* Static HTML Generation

Graphical user interface, text, application, chat or text message

Description automatically generated

- Không cần export any function getStaticProps or getStaticPaths

- Chỉ có HTML không, không có JSON Data, không cần fetch api hay xử lý data gì cả.

\* Static HTML + JSON Data

- Cần 1 số Data để render một số phần nào đấy => sử dụng thêm hàm getStaticProps()

- Phải được export from a Page(các folder hoặc file nằm trong thư mục Page)

- Chỉ được chạy khi build time (phía server side)

- Trong quá trình dev, mỗi khi request thì đều chạy hàm này 1 lần, còn lúc build ra production rồi thì chỉ chạy lúc mình build ra thôi.

- Chạy lại mỗi khi có request được gửi lên trong môi trường dev mode

\* SSG with Dynamic Routes

Graphical user interface, diagram

Description automatically generated with medium confidence

- Flow: nếu có 1 page là posts và bên trong gồm cái bài post [id]. Lúc build nó sẽ đi fetch data từ CMS hoặc từ API nào đó hoặc đọc từ file của system server lên. Từ data đó build ra 1 loạt các file HTML tương ứng. Để làm được việc đó cần sử dụng 1 hàm tên là **getStaticPaths**.

+, Hàm này không sử dụng được chung với hàm **getServerSideProps**

+, Chỉ được chạy khi build time bên phía server-side

+, Phải được export from a **Page**

+, Hàm này return về 1 object có 2 key required là **paths** và **fallback**. **paths** là 1 mảng cái object, các object này sẽ được truyền vào hàm **getStaticProps**, ở **paths** có bao nhiêu item thì sẽ gọi hàm **getStaticProps**() bấy nhiêu lần và tạo ra bấy nhiêu file HTML và JSON tương ứng. **fallback** có 3 giá trị là false, true, “blocking”.

+ id sẽ tuỳ vào [id].js, nếu là [postId] thì sẽ là {param: {postId: ‘1’}}

Text

Description automatically generated

## 4, CSR – Client Side Rendering

- Phân biệt CSR và SSR: 1 cái có prerendering và 1 cái không có prerendering

- Không có pre-rendering: lúc đầu chỉ là 1 file HTML trống rỗng, phải chạy file JS để render lên DOM

- Có pre-rendering: có sẵn DOM ở bên phía server trả về, sau đó chỉ cần hydrate và attach event thôi.

Graphical user interface, text

Description automatically generated

- Khi tạo ra 1 dynamic component, ssr = false => ko sử dụng Server Side Rendering. Component này chỉ chạy bên phía client, không chạy bên phía server

Diagram

Description automatically generated

- Kết hợp SSG và render bên phía client:   
- Ý nghĩa: có những phần layout, muckup style gì đó, phần này render 1 phần bên phía server trước, render ra sẵn 1 cái khung gì đó. Sau khi có khung này rồi, khi load về phía client, đi fetch data bên phía client, fetch xong sẽ render thêm nội dung lên

- Sử dụng thêm 1 useEffect, bên trong callback nhận vào 1 IIFE và xử lý logic, fetch data

## 5, SSR – Server Side Rendering

Diagram

Description automatically generated

- SSR theo kiểu từng request, cứ request lên thì mình đi fetch data tạo ra file HTML và trả về cho user. Cứ mỗi lần request lại fetch tiếp data và liên tục như vậy. Đảm bảo dữ liệu luôn là dữ liệu mới nhưng ảnh hưởng đến performance

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generatedGraphical user interface, text

Description automatically generatedText

Description automatically generated

Text, letter

Description automatically generated

## 6, Incremental Static Regeneration – ISR

- Điểm hạn chế của SSG là chỉ được mỗi khi build time, thay đổi dữ liệu cần build time lại. Với ISR có thể build ra file HTML on runtime.

- ISR có thể hiểu giống với thằng SSR tuy nhiên nó hỗ trợ việc generate ra HTML nghĩa là lúc user request chúng ta có thể render thêm HTML được.

- Có 2 hướng tiếp cận: Faster Builds & Higher Cache Hit Rate

+, Faster Builds: Chọn khoảng 1000 sản phẩm phổ biến để chạy lúc build-time. Nếu user request 1 sản phẩm khác thì generate ra sau. Mất khoảng 1 phút

+, Higher Cache Hit Rate: Muốn user request thì gặp cache nhiều hơn, generate ra 10000 sản phẩm để tỉ lệ user chọn đúng sản phẩm đã generate cao hơn. Mất khoảng 8 phút.

A picture containing graphical user interface

Description automatically generated

- Cách hoạt động

Trong hàm **getStaticProps** thì thêm 1 option là `**revalidate: 60**`, nó sẽ hiểu là chuyển sang dạng ISR.

- Ý nghĩa: Tính từ thời điểm 0s đến mốc config của **revalidate**, nếu user request lên sẽ luôn luôn trả về ngay lập tức từ cache. Tính từ thời điểm **revalidate(60s)** trở đi, nếu như có ai đó request lên, lúc đó vẫn trả về trang cũ, sau đó trigger generate ra 1 version mới và update lên cache. Từ khi update được lên cache rồi, request lên sẽ trả về version mới

- Đối với những đường dẫn không có trong cache, ví dụ có 1000 sản phẩm, user request lên sản phẩm thứ 1001, khi đó option **fallback**  trong hàm **getStaticPaths**

**Fallback : blocking** : khi request lên 1 page chưa có trong cache lúc đó sẽ gọi getStaticProps để generate ra 1 file HTML và return về cho phía client. Và những request sau đó sẽ trả về từ cache. Có nhược điểm là Time to first by khá lâu, phụ thuộc hàm **getStaticProps** chạy nhanh hay chậm

**Fallback: true** : Khi request lên mà trang ko tồn tại, ngay lập tức sẽ update route is fallback, trong lúc này có thể show ra 1 cái loading, khi load xong sẽ update lại state, từ đó lấy dữ liệu và render lên.

## 7. Proxy request to API server

## 8, useSWR

- Tổng quan: là một server state management library để quản lý phần server state (là những state được lưu lại khi gọi api, lấy được dữ liệu và muốn lưu lại dữ liệu đó).

Khi sử dụng useSWR(), gọi tại nhiều chỗ thì nó không trigger nhiều api request mà nó chỉ gọi duy nhất 1 api mà thôi và nó share cho tất cả các nơi sử dụng hook useSWR() này với cùng “key”.

## 9, Font Optimaztion

NextJs sẽ biến phần CSS của Google Font thành inline lúc build time để hạn chế việc fetch request để lấy phần Font Declaration.

Trong lúc “yarn build” Nextjs sẽ đi Fetch phần Font CSS và add inline vào thẻ <style>. Khi Fetch 1 file HTML về thì đã có phần Font rồi ko cần phải tốn 1 request để Fetch phần Font này nữa.

Hiện tại Nextjs chỉ support phần Font Optimazation cho Google Font và Typekit.