Next.js

Khóa học: ReactJS

Mục tiêu

- Nắm được lợi ích của SSR đối với SEO
- Giới thiệu framework Next.js
- Biết được cách điều hướng trang, thêm CSS vào ứng dụng
- Hiểu được cách fetch dữ liệu, dynamic route trong Next.js

Giới thiệu

- Server Side Rendering (SSR) là cơ chế render web trên máy chủ và trả kết quả về cho client và giúp hỗ trợ tốt cho SEO.
- Search Engine Optimization (SEO) là quá trình tăng chất lượng và lưu lượng truy cập website bằng cách tối ưu tệp trích xuất HTML, cấu trúc website thân thiện với các công cụ tìm kiếm dữ liệu như Google, Bing,...
- Next.js là một React framework cung cấp những thành phần cần thiết cho việc tạo các ứng dụng web, trong đó nổi bật là tính năng server side rendering.

Tổng quan về SSR

Khái niệm

- SSR là khả năng của ứng dụng để chuyển đổi các tệp HTML trên máy chủ thành một trang HTML hiển thị đầy đủ nội dung cho máy người dùng.
- Trình duyệt web gửi yêu cầu thông tin từ máy chủ, máy chủ sẽ trả lời ngay lập tức bằng cách gửi một trang chứa đầy đủ thông tin đến máy khách bao gồm HTML, CSS, JavaScript và các nội dung khác cần thiết để hiển thị trang
- Một số framework sử dụng SSR phổ biến Next.js, Nuxt.js và Nest.js.

Tổng quan về SSR

Ưu điểm của SSR

- Thời gian tải trang ban đầu nhanh hơn so với CSR – Điều này mang lại trải nghiệm tốt cho người dùng.
- SEO tốt hơn – Do thời gian tải ban đầu nhanh hơn, các bot của công cụ tìm kiếm có thể thu thập thông tin và lập chỉ mục các trang tốt hơn.
- Tối ưu cho người dùng có kết nối internet chậm – Họ có thể xem trước HTML trong khi JavaScript đang xử lý.

Tổng quan về SSR

Nhược điểm của SSR

- Tốn tài nguyên máy chủ – Trình duyệt gửi yêu cầu đến máy chủ cho mỗi trang.
- Tốn thời gian – Việc xử lí trên máy chủ có thể tốn nhiều thời gian cho việc tiếp nhận yêu cầu, xử lí logic và gửi lại phản hồi.
- Có thể có độ trễ khi tương tác ở thời điểm tải trang – Mặc dù trang HTML hiển thị cho người dùng nhưng họ không thể tương tác với trang đó vì JavaScript được xử lí sau.
- TTFB chậm (Time to first byte) – do cần thời gian để xử lý HTML được kết xuất để phản hồi yêu cầu đầu tiên.

Tổng quan về SSR

So sánh Server-Side Rendering với Client-Side Rendering

Client Side Rendering (CSR) là cơ chế render trang trên trình duyệt của người dùng thay vì trên máy chủ.

Tổng quan về SSR

So sánh Server-Side Rendering với Client-Side Rendering

CSR có một số ưu điểm như:

- Tốn tài nguyên máy chủ – Trình duyệt gửi yêu cầu đến máy chủ cho mỗi trang.
- Thời gian render nhanh – Trang hiển thị nhanh chóng sau thời gian tải trang lần đầu tiên.
- Giảm tải trên máy chủ – Javascript được thực thi trên trình duyệt của khách hàng, tạo ít tải hơn cho máy chủ.

Tổng quan về SSR

So sánh Server-Side Rendering với Client-Side Rendering

CSR tồn tại một số nhược điểm so với SSR:

- Tải chậm – HTML, CSS và Javascript phải được hiển thị trước rồi mới hiển thị cho người dùng, làm tăng thời gian tải trang đầu tiên.
- Gây khó khăn cho SEO – Bot của công cụ tìm kiếm phải đợi tải tất cả các tài nguyên của trang được render hoàn chỉnh.

Tổng quan về SSR

So sánh Server-Side Rendering với Client-Side Rendering

CSR tồn tại một số nhược điểm so với SSR:

- Tải chậm – HTML, CSS và Javascript phải được hiển thị trước rồi mới hiển thị cho người dùng, làm tăng thời gian tải trang đầu tiên.
- Gây khó khăn cho SEO – Bot của công cụ tìm kiếm phải đợi tải tất cả các tài nguyên của trang được render hoàn chỉnh.

Search Engine Optimization (SEO)

Khái niệm

- Tối ưu hóa công cụ tìm kiếm (SEO) là quá trình tăng chất lượng và lưu lượng truy cập website bằng cách tối ưu tệp trích xuất HTML, cấu trúc website thân thiện với các công cụ truy tìm dữ liệu như Google, Bing, Yahoo,...
- Tối ưu Code chuẩn SEO nhằm mục tiêu điều hướng các công cụ tìm kiếm một cách tốt nhất.

Search Engine Optimization (SEO)

Trang website được đánh giá chuẩn SEO thì bao gồm:

- Website thân thiện với người dùng, thiết kế đẹp, nội dung tương tác tốt với người dùng
- Website có cấu trúc thân thiện với các công cụ tìm kiếm, giúp việc tìm kiếm dễ dàng thu thập thông tin.
- Phần quản trị website có đầy đủ các cơ chế quản trị thân thiện dễ dàng tùy biến SEO.

Search Engine Optimization (SEO)

Các loại hình SEO gồm có:

- SEO từ khóa là loại hình thông dụng nhất hiện nay. Nhà quản trị dựa trên SEO từ khóa tiếng Việt có dấu và không dấu, nhằm mục đích cải thiện thứ hạng của website trên công cụ tìm kiếm, tăng khả năng tiếp cận với người dùng.
- SEO hình ảnh là đưa hình ảnh trên website ưu tiên hiển thị ở những vị trí đầu trên trang tìm kiếm hình ảnh của những công cụ như Google.

Search Engine Optimization (SEO)

Các loại hình SEO gồm có:

- SEO video social là nâng cao thứ hạng tìm kiếm của website nhờ các trang mạng xã hội và tương tác người dùng.
- SEO app mobile sẽ đưa ứng dụng của doanh nghiệp trên appstore hay google play giúp người dùng dễ dàng tìm thấy và tải ứng dụng.

Lợi ích SSR đối với SEO

- Việc render trên máy chủ sẽ tăng tốc thời gian tải trang, điều này không chỉ cải thiện trải nghiệm người dùng mà còn có thể giúp trang web của bạn xếp hạng tốt hơn trong kết quả tìm kiếm của Google.
- SSR tốt hơn cho SEO vì nó loại bỏ gánh nặng hiển thị JavaScript khỏi bot của công cụ tìm kiếm, giải quyết các vấn đề liên quan đến thu thập dữ liệu, tốc độ.

Next.js

Để xây dựng một ứng dụng web hoàn chỉnh với React từ đầu, có nhiều chi tiết quan trọng bạn cần xem xét:

- Code phải được đóng gói bằng webpack và được chuyển đổi bằng trình biên dịch như Babel.
- Bạn có thể muốn render trước một số trang cho SEO.
- Bạn muốn thực hiện tối ưu hóa cho trang web chạy nhanh bằng tách mã.
- Bạn cũng có thể muốn render phía máy chủ hoặc render phía máy khách.
- Bạn có thể phải viết một số mã phía máy chủ để kết nối ứng dụng React với kho dữ liệu của bạn.

Next.js

Và Next.js sẽ cung cấp giải pháp cho tất cả các vấn đề trên.

- Next.js là một React framework, nó cung cấp những thành phần cần thiết để giúp bạn tạo ra các ứng dụng web một cách nhanh chóng, trong đó phải kể đến tính năng server side rendering và tạo ra các trang web tĩnh.

Next.js

Tính năng của Next.js

- Pre-rendering, static generation (SSG) và server-side rendering (SSR)
- Tách mã tự động để tải trang nhanh hơn, chỉ load những gì cần thiết cho mỗi page .
- Hỗ trợ css và sass, hỗ trợ bất kì thư viện css trong JS
- Hỗ trợ refresh page nhanh chóng ở môi trường development
- Môi trường phát triển và hỗ trợ làm mới nhanh
- Các routerAPI xây dựng các điểm cuối API với chức năng không có máy chủ
- Có thể mở rộng

Next.js

Từ đó ta có thể thấy Next.js có nhiều lợi ích như:

- Cải thiện quy trình phát triển bằng chi phí và thời gian mang lại lợi ích cho khách hàng
- Cải thiện hiệu suất bằng ứng dụng nhanh hơn
- Cải thiện SEO bằng các ứng dụng thân thiện với SEO

Next.js

Cài đặt Next.js

- npx create-next-app nextjs-demo

Chạy development server

- npm run dev

Điều hướng giữa các trang

Trong Next.js, một trang là một React Component được xuất từ một file trong thư mục pages. Các trang được liên kết với một route dựa trên tên file của chúng. Ví dụ:

- pages/index.js được liên kết với route “/“.
- pages/posts/first-post.js được liên kết với route “/posts/first-post”.

Điều hướng giữa các trang

Cách tạo một trang

- Tạo file JS bên trong thư mục pages
- Tạo component với export default

```
export default function FirstPost() {

  return <h1>First Post</h1>;

}
```

Điều hướng giữa các trang

Link Component

- Link component cho phép bạn thực hiện điều hướng phía máy khách đến một trang khác trong ứng dụng.
- Import Link component từ “next/link”
- Link component tương tự như sử dụng thẻ <a>, nhưng thay vì <a href=”…”>, bạn sử dụng <Link href = “…”> và đặt thẻ <a> bên trong.

Điều hướng giữa các trang

Client-Side Navigation

- Client-Side Navigation có nghĩa là quá trình chuyển đổi trang diễn ra bằng JavaScript, nhanh hơn điều hướng mặc định do trình duyệt thực hiện.
- Next.js thực hiện phân tách mã tự động, vì vậy mỗi trang chỉ tải những gì cần thiết cho trang đó. Điều này đảm bảo rằng trang chủ tải nhanh chóng ngay cả khi bạn có hàng trăm trang.
- Bạn sẽ tạo route dưới dạng tệp dưới các trang và sử dụng Link component, mà không cần các thư viện route.

CSS Styling

CSS trong Next.js

- Next.js có hỗ trợ tích hợp cho kiểu JSX, nhưng bạn cũng có thể sử dụng các thư viện CSS-in-JS phổ biến khác như styled-components hoặc emotion.
- Styled-jsx. Đó là thư viện “CSS-in-JS” – nó cho phép bạn viết CSS trong React component và các kiểu CSS sẽ được xác định phạm vi (các thành phần khác sẽ không bị ảnh hưởng).
- Next.js có hỗ trợ tích hợp cho CSS và Sass, cho phép bạn nhập các tệp .css và .scss.

CSS Styling

Tạo Layout Component

- Layout component có nhiệm vụ tạo layout chung cho các trang.
- Code tạo component layout

```
export default function Layout({ children }) {

    return <div>{children}</div>;

}
```

CSS Styling

Sử dụng Layout Component

```
export default function FirstPost() {
  return (
    <Layout>
      <Head>
        <title>First Post</title>
      </Head>
      <h1>First Post</h1>
      <h2>
        <Link href="/">
          <a>Back to home</a>
        </Link>
      </h2>
    </Layout>
  );
}
```

CSS Styling

Thêm style cho Layout Component

- Mô-đun CSS, cho phép bạn nhập file CSS trong một React component.
- Tạo một file CSS có tên là layout.module.css. Để sử dụng mô-đun CSS, tên file phải được kết thúc với .module.css
- Sử dụng style

```
import styles from "./layout.module.css";
export default function Layout({ children }) {
  return <div className={styles.container}>{children}</div>;
}
```

CSS Styling

Global style

- Khi bạn muốn CSS được load cho mọi trang, Next.js sẽ hỗ trợ bạn với CSS global.
- Bạn có thể đặt file global CSS ở bất kỳ đâu và sử dụng bất kỳ tên nào.
- Thêm các tệp global CSS bằng cách import chúng trong pages/_app.js.

```
import '../styles/global.css'

export default function App({ Component, pageProps }) {
  return <Component {...pageProps} />
}
```

Data Fetching

Khái niệm Pre-rendering

- Next.js sẽ render lại ở mỗi trang, nghĩa là HTML sẽ được khởi tạo ở mỗi page thay vì sử dụng javascript để xử lý phía client. Pre-rendering sẽ tốt hơn cho performance và SEO

Data Fetching

Hai hình thức Pre-rendering là:

- Static Generation: HTML được khởi tạo ngay tại thời điểm build app (npm run build). HTML được render trước và sử dụng lại theo từng request
- Server-side Rendering: khởi tạo HTML trên mỗi request

Data Fetching

Nên sử dụng hình thức nào?

- Sử dụng Static Generation (có và không có dữ liệu) bất cứ khi nào có thể vì trang của bạn được tạo một lần và được CDN phân phát, điều này làm cho nó nhanh hơn nhiều so với việc máy chủ hiển thị trang theo mọi yêu cầu.
- Nếu trang của bạn cần cập nhật dữ liệu thường xuyên, nội dung trang thay đổi trên mỗi request, hãy sử dụng Server Side Rendering.
- Next.js cho phép bạn tuỳ chọn kiểu pre-rendering cho mỗi trang.

Data Fetching - Cách lấy dữ liệu Next.js

getStaticProps (Static Generation)

- Phương thức getStaticProps có thể được sử dụng bên trong một Page để lấy dữ liệu ngay tại thời điểm xây dựng. Một khi ứng dụng đã được xây dựng, nó sẽ không làm mới dữ liệu cho đến khi một bản dựng khác được khởi động.

```
export async function getStaticProps(context) {
  // fetch dữ liệu từ file system, API, DB,...
  const data = ...

  // Giá trị của `props` sẽ được truyền tới component `Home`
  return {
    props: ...
  }
}

export default function Home(props) { ... }
```

Data Fetching - Cách lấy dữ liệu Next.js

getServerSideProps (Server-side Rendering)

- Phương thức này lấy dữ liệu mỗi khi user gửi request lên hệ thống. Nó sẽ tìm nạp dữ liệu trước khi client có thể view được trang web. Nếu client đưa ra các request tiếp theo, dữ liệu sẽ được tìm và nạp lại.

```
export async function getServerSideProps(context) {
  return {
    props: {
      // Giá trị props cho component của bạn
    },
  };
}
```

Routing

Giới thiệu

- Next.js có một hệ thống route được xây dựng dựa trên khái niệm về các trang.
- Khi một tệp được thêm vào thư mục pages, tệp đó sẽ tự động có sẵn dưới dạng một route.

Routing

Index routes

- Là route sẽ tự động định tuyến các tệp đến thư mục gốc.

  pages/index.js → /

  pages/blog/index.js → /blog

Routing

Nested routes

- Là route hỗ trợ các tệp được lồng vào nhau. Nếu bạn tạo cấu trúc thư mục lồng nhau, các tệp sẽ tự động được định tuyến theo cùng một cách.

  pages/blog/first-post.js → /blog/first-post

  pages/dashboard/settings/username.js → /dashboard/settings/username

Routing

Dynamic routes

- Dynamic route là các route cho phép chúng ta thêm các tham số tuỳ chọn vào URL. Next.js sử dụng dấu ngoặc [] trong tên của file, điều này cho phép bạn so khớp các tham số được đặt tên:
- pages/blog/[slug].js → /blog/:slug (/blog/hello-world)
- pages/[username]/settings.js → /:username/settings (/foo/settings)
- pages/post/[...all].js → /post/* (/post/2020/id/title)

Dynamic Route

Cách tạo

- Trong Next.js, bạn có thể thêm dấu ngoặc vào một trang ([param]) để tạo dynamic route
- Cùng xem page sau: pages/post/[pid].js

```
import { useRouter } from 'next/router'
const Post = () => {
  const router = useRouter()
  const { pid } = router.query
  return <p>Post: {pid}</p>
}

export default Post
```

Dynamic Route

Thông qua ví dụ trên ta sẽ phân tích

- Bất kỳ route nào như /post/1, /post/abc, v.v. sẽ được khớp với các pages/post/[pid].js.
- Ví dụ: route/post/abc sẽ có đối tượng truy vấn sau: {"pid": "abc"}
- Tương tự, route/post/abc?foo=bar sẽ có đối tượng truy vấn sau: {"foo": "bar", "pid": "abc"}
- Tuy nhiên, các route parameter sẽ ghi đè các tham số truy vấn có cùng tên. Ví dụ: route /post/abc?pid=123 sẽ có đối tượng truy vấn sau: {"pid": "abc"}

Dynamic Route

Thông qua ví dụ trên ta sẽ phân tích

- Nhiều dynamic route hoạt động theo cùng một cách. Các trang của pages/post/[pid]/[comment].js sẽ khớp với route /post/abc/a-comment và đối tượng truy vấn của nó sẽ là:

  {"pid": "abc", "comment": "a-comment"}

Dynamic Route

Sử dụng Client-side navigations với dynamic route, next/link

```
import Link from 'next/link'

function Home() {
  return (
    <ul>
      <li>
        <Link href="/post/abc">
          <a>Go to pages/post/[pid].js</a>
        </Link>
      </li>
      <li>
        <Link href="/post/abc?foo=bar">
          <a>Also goes to pages/post/[pid].js</a>
        </Link>
      </li>
      <li>
        <Link href="/post/abc/a-comment">
          <a>Go to pages/post/[pid]/[comment].js</a>
        </Link>
      </li>
    </ul>
  )
}

export default Home
```

Dynamic Route

Link tới dynamic path

Ví dụ hiển thị danh sách các post kèm theo là dynamic link xem chi tiết post

Dynamic Route

Link tới dynamic path - Sử dụng định dạng chuỗi (string interpolation)

```
function Posts({ posts }) {
  return (
    <ul>
      {posts.map((post) => (
        <li key={post.id}>
          <Link href={`/blog/${encodeURIComponent(post.slug)}`}>
            <a>{post.title}</a>
          </Link>
        </li>
      ))}
    </ul>
  )
}
```

Dynamic Route

Link tới dynamic path - Sử dụng Object URL

```
function Posts({ posts }) {
  return (
    <ul>
      {posts.map((post) => (
        <li key={post.id}>
          <Link
            href={{
              pathname: '/blog/[slug]',
              query: { slug: post.slug },
            }}
          >
            <a>{post.title}</a>
          </Link>
        </li>
      ))}
    </ul>
  )
}
```

Tóm tắt bài học

- Nắm được lợi ích của SSR đối với SEO
- Giới thiệu framework Next.js
- Biết được cách điều hướng trang, sử dụng CSS với Next.js
- Hiểu được cách fetch dữ liệu, Cơ chế routing, dynamic route trong Next.js