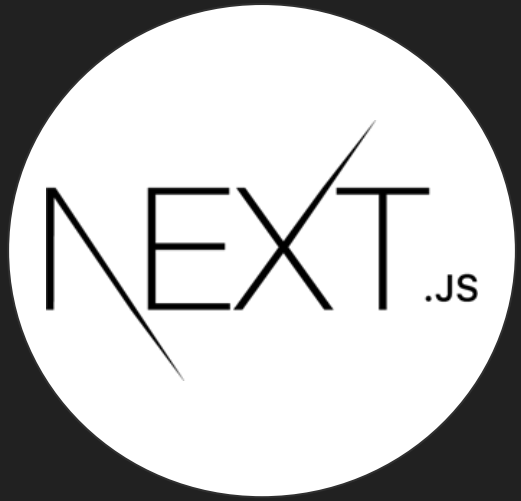
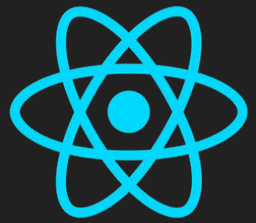


รู้จักกับ Next.js

https://www.youtube.com/c/KongRuksiamOfficial/

https://www.facebook.com/KongRuksiamTutorial/

Next.js คืออะไร



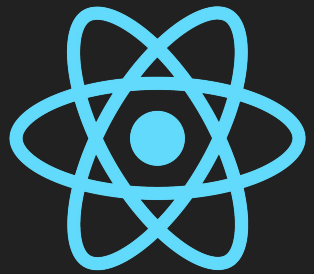
เป็น React Web Framework เพื่อแก้ปัญหาสำหรับเว็บที่ต้องการทำ SEO (Search Engine Optimization) คือทำให้เว็บติดหน้าแรกของ Search Engine เป็นการทำให้เว็บค้นหาผ่าน Google เจอได้ง่ายมากยิ่งขึ้น เนื่องจาก Next.js นั้นช่วยทำ SSR (Server Side Rendering)

ข้อดีของ Next.js

- Server Side Rendering (SSR)
- Static Site Generation (SSG) ทำการ build และ Render HTML จากฝั่ง Server แล้วส่ง HTML ไปทำงานที่ฝั่ง Client
- Hot Reload (แก้ไขการทำงานแล้วเห็นผลทันที)
- Routing สามารถจัดการ Routing ง่ายๆผ่าน File System (Pages)
- Optimization จัดการเรื่อง Performance (ทำให้เว็บทำงานเร็วขึ้น)

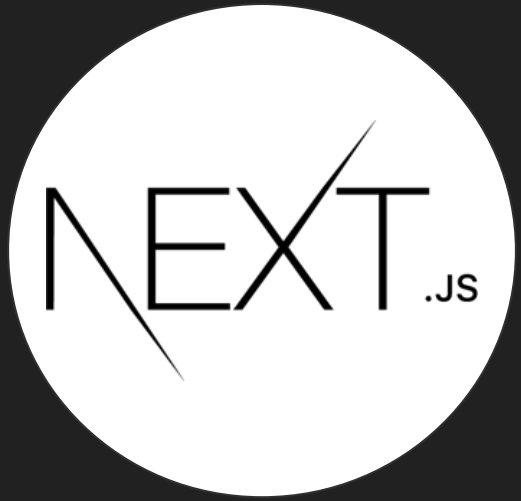
ต้องมีพื้นฐานอะไรบ้าง

- HTML5
- CSS3
- JavaScript
- React เบื้องต้น



เครื่องมือ

- Visual Studio Code
- Node.js
- Google Chrome



รู้จักกับ CSR , SSR และ SSG

https://www.youtube.com/c/KongRuksiamOfficial/

https://www.facebook.com/KongRuksiamTutorial/

รู้จักกับ CSR , SSR และ SSG

- Client Side Rendering (CSR)
- Server Side Rendering (SSR)
- Static Site Generation (SSG)

รู้จักกับ CSR , SSR และ SSG

- Client Side Rendering (CSR)
- Server Side Rendering (SSR)
- Static Site Generation (SSG)

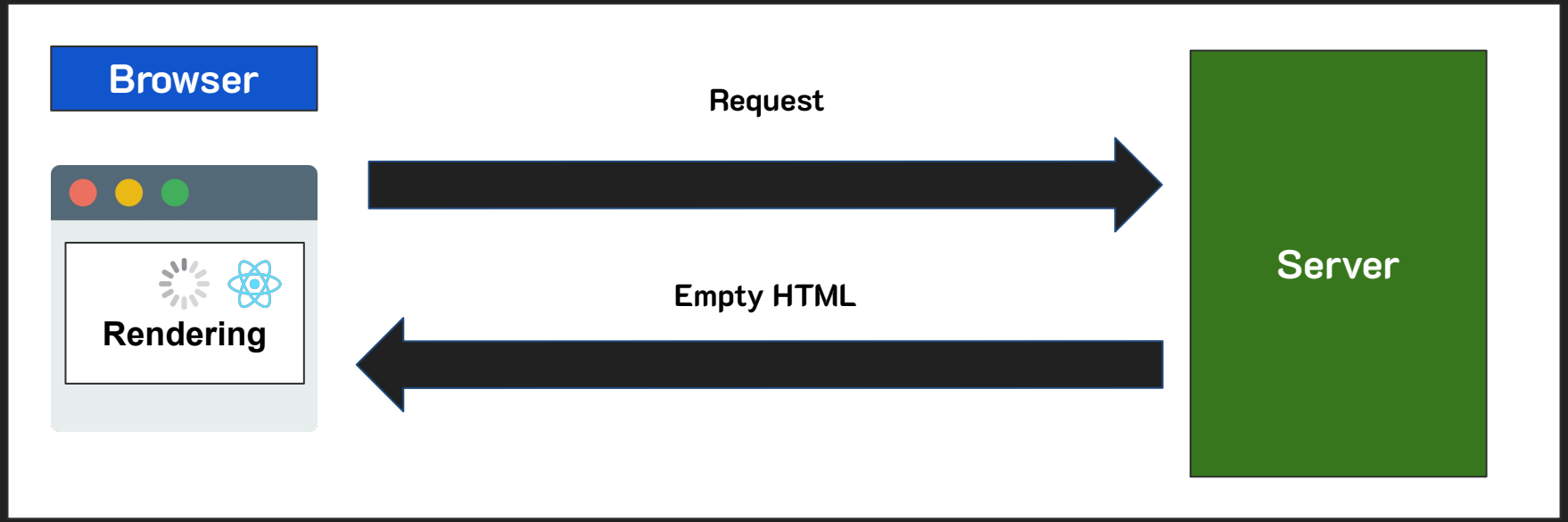
CSR (Client Side Rendering)

CSR (Client Side Rendering) กระบวนการทำงานทั้งหมดจะเกิดขึ้นที่ฝั่งผู้ใช้ (Client) ตั้งแต่การโหลดหน้าเว็บและ Execute JavaScript โดยอาศัยทรัพยากรเครื่องผู้ใช้ทั้งหมด ซึ่งจะรับไฟล์ HTML มาก่อนแล้วจึงจะมีการโหลดเนื้อหาอื่น ๆ ตามมาในภายหลัง

CSR (Client Side Rendering)

จากนั้นจะเป็นหน้าที่ของ Framework/Library
ที่จะ Render HTML และควบคุมการทำงานของเว็บ
ไซต์จาก Browser ของผู้ใช้งาน

CSR (Client Side Rendering)



ข้อดีของ CSR

- ลดต้นทุนด้าน Server เนื่องจากขั้นตอนการประมวลผลเว็บไซต์มาอยู่ที่ Web Browser ของฝั่งผู้ใช้งาน
- รองรับ Single Page Application
- รองรับ Static Website สามารถทำงานได้เลยโดยไม่จำเป็นต้องพึ่งพากันรันแอพพลิเคชั่นผ่าน Server

ข้อเสียของ CSR

- หากเว็บไซต์มีขนาดใหญ่ก็จะใช้เวลาโหลดข้อมูลนาน ส่งผลต่อประสบการณ์ของผู้ใช้ในการใช้งานเว็บ
- ส่งผลต่ออันดับเว็บไซต์ (SEO) ที่ขึ้นมาบน Search Engine เนื่องจากมองเห็นเนื้อหาบนเว็บไซต์ได้ยากกว่าแบบ SSR (Server Side Rendering)

รู้จักกับ CSR , SSR และ SSG

- Client Side Rendering (CSR)
- Server Side Rendering (SSR)
- Static Site Generation (SSG)

รู้จักกับ CSR , SSR และ SSG

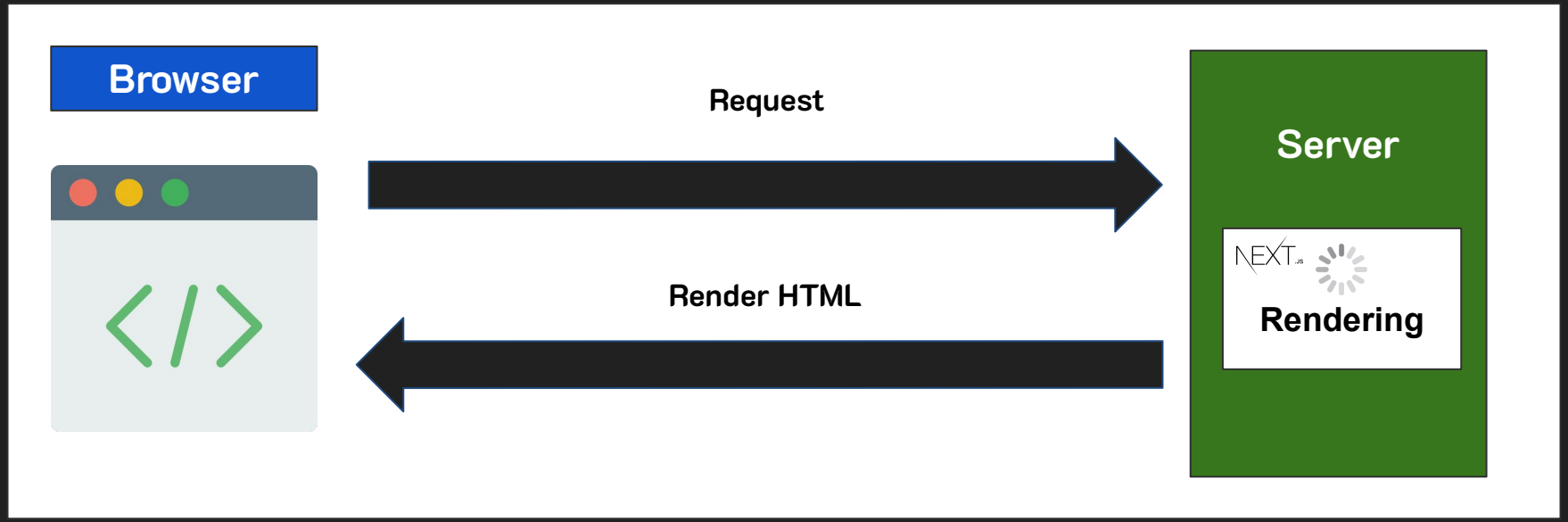
- Client Side Rendering (CSR)
- Server Side Rendering (SSR)
- Static Site Generation (SSG)

SSR (Server Side Rendering)

SSR (Server Side Rendering) กระบวนการทำงานทั้งหมดจะย้ายไปที่ฝั่ง Server โดยจะทำการ Render หน้าเว็บบน Server ก่อนที่จะส่งมาให้ผู้ใช้งาน (Client)

หมายความว่าหน้าเว็บที่ส่งมาให้กับผู้ใช้นั้นพร้อมใช้งานแล้ว เว็บเบราว์เซอร์สามารถนำไปแสดงผลได้ทันที ทำให้การใช้งานเว็บไซต์นั้นมีความรวดเร็วมากยิ่งขึ้น

SSR (Server Side Rendering)



ข้อดีของ SSR

- หน้าเว็บสามารถพร้อมใช้งานทันทีได้ เนื้อหาครบถ้วนตั้งแต่ Request แรก ทำให้ผู้ใช้งานสามารถเห็นเนื้อหาของเว็บไซต์ทั้งหมด โดยไม่จำเป็นต้องรอโหลดทีละส่วน
- ช่วยเรื่อง SEO เนื่องจากมีการแสดงผลเนื้อหาในหน้าเว็บทั้งหมดไว้สำหรับ Search Engine Bot

ข้อเสียของ SSR

- เพิ่มต้นทุนด้าน Server เนื่องจากขั้นตอนการประมวลผลเว็บไซต์ทั้งหมดย้ายมาอยู่ที่ฝั่ง Server
- ไม่สามารถทำงานกับ JavaScript Library หรือ Framework บางตัวได้

ข้อเสียของ SSR

- เนื่องจาก Server นั้นต้องมีการตอบสนองกับ Request ของผู้ใช้งาน ซึ่งการส่ง Request แต่ละครั้งจะมีการประมวลผลหรือ Render เว็บไซต์ทั้งหน้า ดังนั้นหากมีการเรียกใช้มากเกินไปจะส่งผลให้ Server ล่มได้

รู้จักกับ CSR , SSR และ SSG

- Client Side Rendering (CSR)
- Server Side Rendering (SSR)
- Static Site Generation (SSG)

รู้จักกับ CSR , SSR และ SSG

- Client Side Rendering (CSR)
- Server Side Rendering (SSR)
- Static Site Generation (SSG)

SSG (Static Site Generation)

SSG (Static Site Generation) มีลักษณะคล้ายกับ SSR (Server Side Rendering) แต่กระบวนการทำงานของ SSG นั้นจะ Render HTML ล่วงหน้าไว้เรียบร้อยตั้งแต่ตอน Build และพร้อมนำไปใช้งานได้ทันที แต่จะไม่ Render ทุกๆ Request เหมือนกัน SSR

SSG (Static Site Generation)

กล่าวคือ SSG นั้นจะสร้างไฟล์ HTML เตรียมรอไว้ เมื่อ Build เสร็จเรียบร้อย ก็จะได้ชุดของไฟล์ HTML ที่สามารถนำไปทำงานแบบเดียวกับ Client Side Rendering (CSR) โดยที่ไม่จำเป็นต้องรันแอพพลิเคชั่นไว้รับ Request ตลอดเวลานั่นเอง (ไม่มี Server)

ข้อดีของ SSG

- เนื่องจากหน้าเว็บมีข้อมูลต่าง ๆ ครบถ้วนอยู่แล้ว ไม่จำเป็นต้องมี Server ไว้เก็บข้อมูล จึงมีความปลอดภัยสูง
- ประหยัดค่าใช้จ่ายด้าน Server เพราะไม่มีการ Render ที่ฝั่ง Server เลย การทำงานมีแค่ส่งไฟล์หน้าเว็บไปที่ฝั่งผู้ใช้งาน และแสดงผลหน้าเว็บในไฟล์ดังกล่าว

ข้อเสียของ SSG

- เมื่อมีการอัพเดตกระบวนการทำงานในเว็บไซต์ต้อง Build หน้าเว็บใหม่ทั้งหมด เนื่องจากไม่มี Server มารองรับการอัพเดตกระบวนการทำงานดังกล่าว
- หากเว็บไซต์มีเนื้อหามากหรือมีขนาดใหญ่ก็จะส่งผลต่อระยะเวลาในการ Build ด้วย

ติดตั้ง Node.js

ติดตั้ง Visual Studio Code

สร้างโปรเจกต์ Next.js

สร้างโปรเจกต์ Next.js

```
npx create-next-app <ชื่อโปรเจกต์>
```

รันโปรเจกต์

- cd ชื่อโปรเจกต์
- npm run dev

โครงสร้างโปรเจกต์

โครงสร้างโปรเจกต์

โครงสร้างโปรเจกต์

- pages คือ โฟลเดอร์ที่ใช้เก็บไฟล์หน้าเว็บเพจหรือส่วนที่แสดงผลต่างๆ ภายในเว็บไซต์ของเรา การสร้างโฟลเดอร์และไฟล์จะมีผลต่อการกำหนดเส้นทาง (Routing) หรือ URL ของเว็บ

  โดย Next.js นั้นจะจัดการ Route ผ่านโฟลเดอร์ pages และสร้างไฟล์ .js ก็สามารถเข้า Route ตามชื่อไฟล์ที่สร้างได้เลย (ยกเว้น index.js ไม่ต้องอ้างอิง)

โครงสร้างโปรเจกต์

- public คือ โฟลเดอร์ที่ใช้จัดเก็บไฟล์ที่นำไปใช้ในโปรเจกต์ เช่น ไฟล์รูปภาพ เป็นต้น
- styles คือ โฟลเดอร์ที่เก็บไฟล์ CSS

โครงสร้างโปรเจกต์

- next.config.js คือ ไฟล์ที่ใช้สำหรับตั้งค่าหรือกระบวนการทำงานในโปรเจกต์
- _app.js คือ ไฟล์เพจหลักในการรันแอพพลิเคชั่น โดยนำส่วนประกอบต่างๆ (Component)มาประกอบกันแล้วนำไปแสดงผลในบราวเซอร์

Pages & Routing

Pages & Routing

- page คือ ไฟล์หน้าเว็บเพจหรือส่วนที่แสดงผลต่างๆ ภายในเว็บไซต์ การสร้างโฟลเดอร์และไฟล์จะมีผลต่อการกำหนดเส้นทาง (Routing) หรือ URL ของเว็บ

ตัวอย่าง URL	Pages
https://localhost:3000/	pages / index.js
https://localhost:3000/about	pages/about.js
https://localhost:3000/products/	pages/products/index.js

Next Link

https://www.youtube.com/c/KongRuksiamOfficial/

https://www.facebook.com/KongRuksiamTutorial/

Next Link

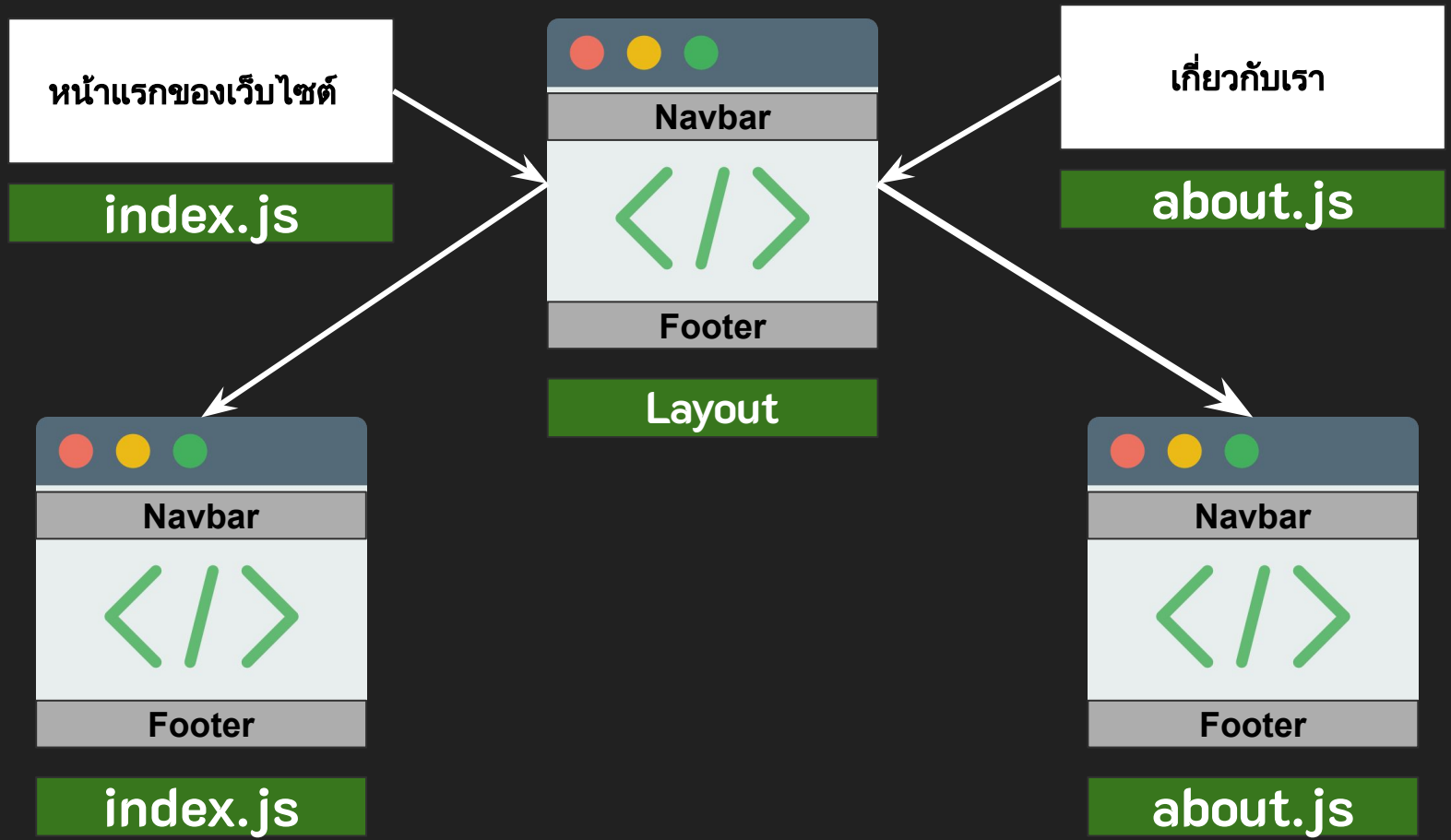
ถ้าต้องการอยากให้หน้าเว็บเพจแต่ละหน้ามีการเชื่อมโยงกัน ใน Next.js จะใช้ Link Component

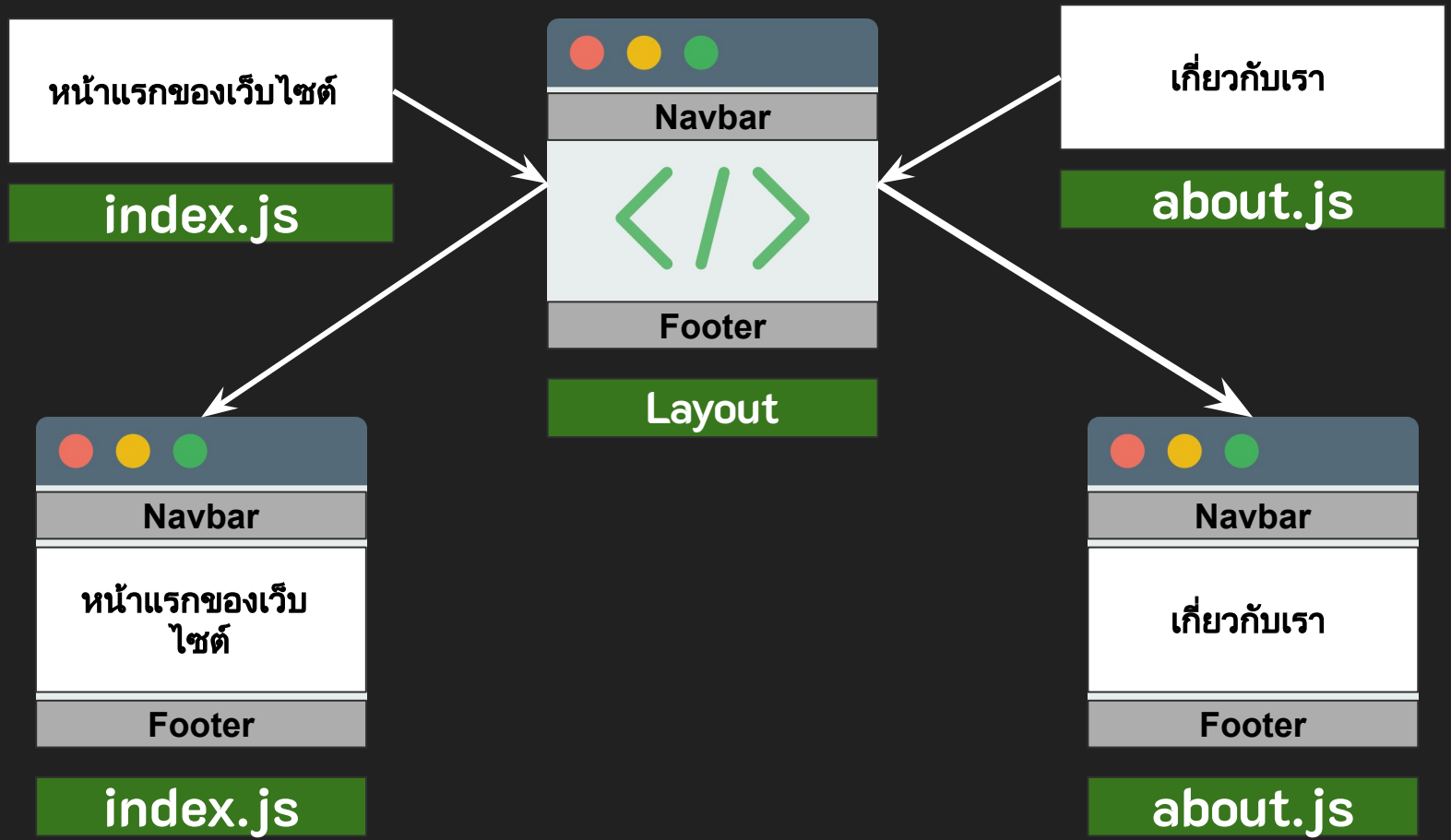
- import Link from 'next/link'
- <Link href="/">HomePage</Link>

Layout Component

Layout Component

คือ การกำหนดโครงสร้างหลักในหน้าเว็บเพจที่ทุกๆหน้า ใช้งานร่วมกันเพื่อลดความซ้ำซ้อนของโค้ด เช่น Navbar และ Footer เป็นต้น


หน้าแรกของเว็บไซต์
index.js
Navbar
Footer
Layout
เกี่ยวกับเรา
about.js
Navbar
Footer
index.js
Navbar
Footer
about.js


หน้าแรกของเว็บไซต์
index.js
Navbar
Footer
Layout
เกี่ยวกับเรา
about.js
Navbar
หน้าแรกของเว็บไซต์
Footer
index.js
Navbar
เกี่ยวกับเรา
Footer
about.js

Layout Component

Children ถูกนำมาทำงานร่วมกับการแสดงผลในโครงสร้างหลัก (Layout) ที่ทำงานในแอพพลิเคชั่น แต่จะบรรจุเนื้อหาด้านในแตกต่างกัน ส่งผลให้ Component มีความแตกต่างหลากหลายมากขึ้น

Next Image

Next Image

ถ้าต้องการอยากให้หน้าเว็บเพจแต่ละหน้ามีการใช้งานรูปภาพ ใน Next.js จะใช้ Image Component

- import Image from 'next/image'
- <Image src="ตำแหน่งภาพ" width={กว้าง} height={สูง} alt="logo"/>

CSS Style ใน Next.js

- ถ้าต้องการอยากให้ทุกๆ Component มีการใช้งาน CSS Style แบบเดียวกัน จะต้องเขียนคำสั่ง CSS ในไฟล์ที่ชื่อว่า global.css

- ถ้าต้องการให้ค่า CSS Style นั้นมีผลเฉพาะ Component ที่สนใจจะต้องเขียนในรูปแบบ Component Style (CSS Module) โดยมีโครงสร้างไฟล์คือ [name].module.css

ดึงข้อมูลจาก API

ภาพรวมโปรเจกต์

- สร้างโปรเจกต์ในรูปแบบ SSG : static HTML + JSON
- API ที่ใช้งาน คือ https://dummyjson.com/
- ดึงข้อมูลสินค้าทั้งหมดมาแสดงผล
- ดูรายละเอียดสินค้าแต่ละรายการได้

การดึงข้อมูลจาก API

getStaticProps ซึ่งเป็น Static Site Generation ตัว Next.js สำหรับดึงข้อมูลจาก API ที่ต้องการนำมาใช้งานและส่งข้อมูล (props) ไปใช้งานใน Component

โดยจะทำงานเพียงครั้งเดียวเมื่อเกิดการ Build ซึ่งจะใช้กับหน้าเว็บที่ต้องการแสดงข้อมูลทั้งหมดโดยที่ข้อมูลดังกล่าวไม่มีการเปลี่ยนแปลง

โครงสร้างคำสั่ง

```
export async function getStaticProps(){
    const res = await fetch(API_URL);
    const data = await res.json();
    return {
        props: { propsName : data}
    }
}
```

getStaticPaths

getStaticPaths

getStaticPaths คือ การใช้งานไดนามิก Routing เพื่อเก็บข้อมูล Path ที่ใช้งานร่วมกับ API (getStaticPaths จะใช้งานร่วมกับ getStaticProps เสมอ)

โดยจะทำงานเพียงครั้งเดียวเมื่อเกิดการ Build ซึ่งจะใช้กับหน้าเว็บที่ต้องการแสดงข้อมูลแบบรายการเดียว โดยที่ข้อมูลดังกล่าวไม่มีการเปลี่ยนแปลง

โครงสร้างคำสั่ง

```
export async function getStaticPaths() {
    return {
      paths: [{ params: { id: 1 }}, { params: { id: 2 }}],
      fallback: false //ถ้าหา path ไม่เจอให้ return 404 Page
    }
}
```