Written by Dr. Thapanapong Rukkanchanunt

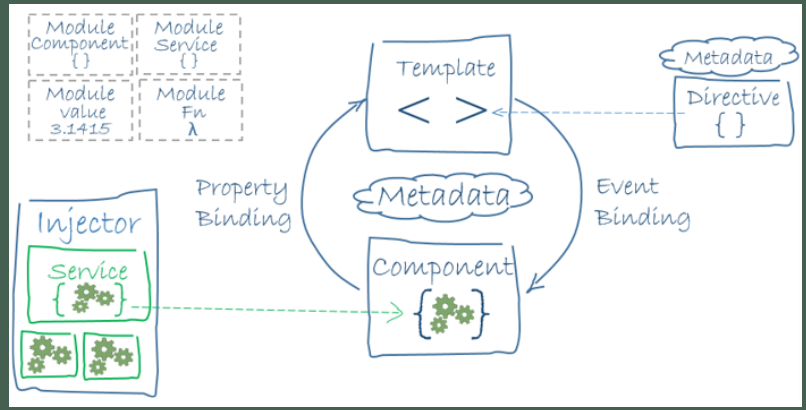
08 Web Application

OUTLINE

- Angular Architecture
  - Angular Module and Library
  - Angular Component
  - Angular Services
  - Routing
- Beyond Web Application
  - New Facebook

Web Application Design Principles

- หนึ่งในสิ่งสำคัญของการสร้างเว็บแอพ คือ การออกแบบโครงสร้างภายใน
- การออกแบบจะขึ้นอยู่กับ Framework ที่ใช้ เนื่องจากหลักการออกแบบแตกต่างกัน
- ในคลาสนี้เราจะโฟกัสไปที่หลักการออกแบบ Angular

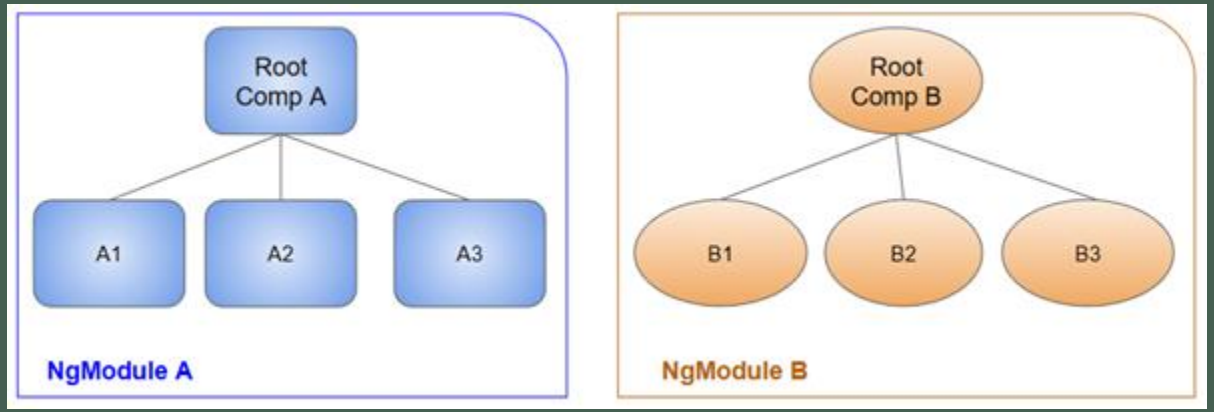


Angular Architecture

- Angular เป็น Framework ที่ใช้สำหรับสร้าง Single Page Application
- Building Block ของ Angular คือ NgModules ซึ่งเป็นตัวที่ Compile แต่ละ Component
- Angular App จึงประกอบไปด้วยหลาย NgModules แต่จะมี Root Module ที่ควบคุม Bootstrap
- Component เป็นส่วนที่กำหนดการแสดงผลของหน้าจอ Component ใช้ Service ที่ให้บริการฟังก์ชันต่าง ๆ ที่ไม่เกี่ยวข้องกับการแสดงผล
- ทั้ง Component และ Service คือคลาส แต่จะมี Decorator เป็นตัวกำหนดหน้าที่
- Component อาจจะแยกออกเป็นการแสดงผลส่วนย่อย ๆ ในหน้าจอ แต่จะมี Router เป็นตัวเชื่อมการแสดงผลระหว่างส่วนย่อย

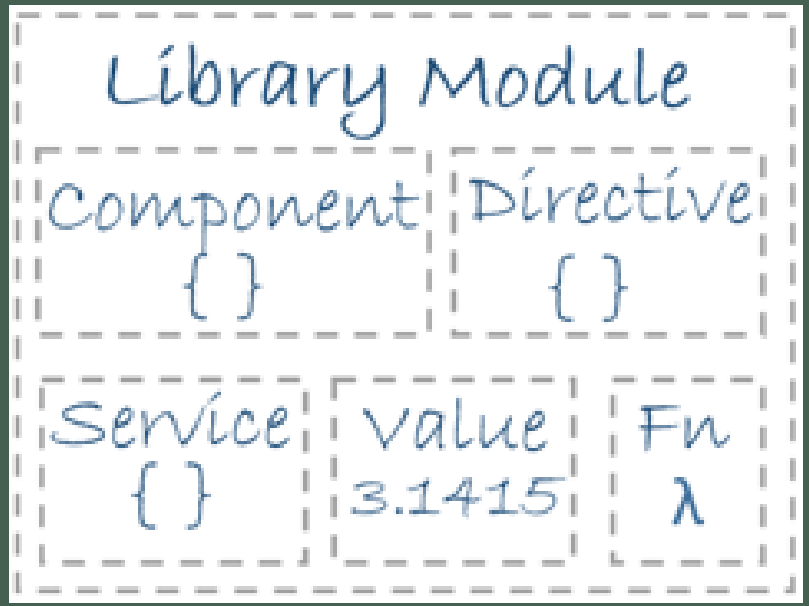
Angular Modules

- NgModule เปรียบเสมือน Operating System สำหรับเว็บแอพ เป็นโค้ดที่เอาไว้ Compile Component และ Service ต่าง ๆ ที่ประกาศภายใน NgModule นั้น
- Root Component ใน NgModule จะเป็น Component แรกที่ถูกสร้างตอนเริ่มเว็บแอพ
- Component อื่น ๆ จะถูกสร้างเมื่อมีการ Route ไปหา (ผ่าน Tag หรือ Router Module)



Angular Libraries

- Angular มี Angular Library Module ที่เราสามารถ Import ไปยัง Module อื่นได้
- ชื่อของ Library จะขึ้นต้นด้วย @angular



Angular Component

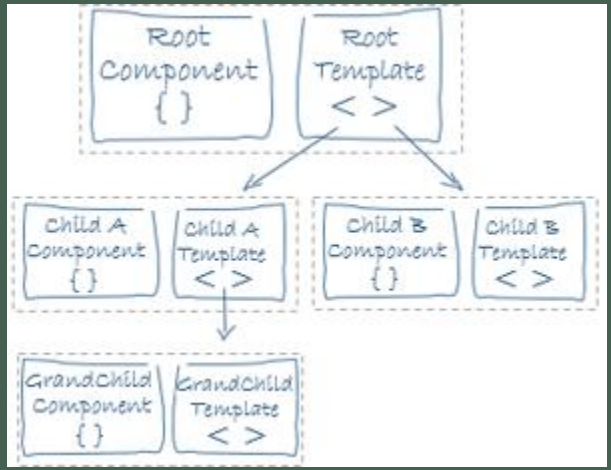
- Root Component เชื่อมโยง Component Hierarchy กับ Document Object Model (DOM) โดยใน Component จะต้องการประกาศคลาสที่ประกอบไปด้วย ข้อมูลในแอพ และ ตรรกะ
- Decorator ชื่อ @Component() เป็นตัวกำหนดว่าคลาสนี้จะทำหน้าที่เป็น Component

```
import { Component, OnInit } from '@angular/core';
import { Router } from '@angular/router';
import { FormBuilder, FormGroup, Validators} from '@angular/forms';
import { AuthService } from '../auth.service';

@Component({
  selector: 'app-login',
  templateUrl: './login.component.html',
  styleUrls: ['./login.component.css']
})
export class LoginComponent implements OnInit {
```

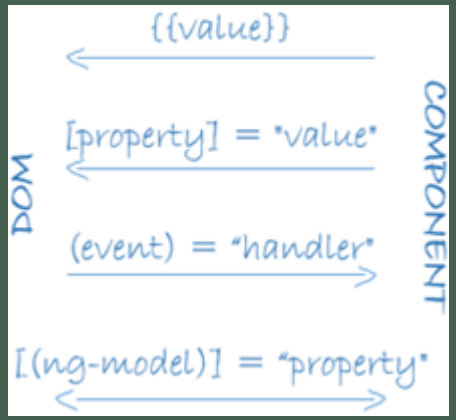
Templates and Directive

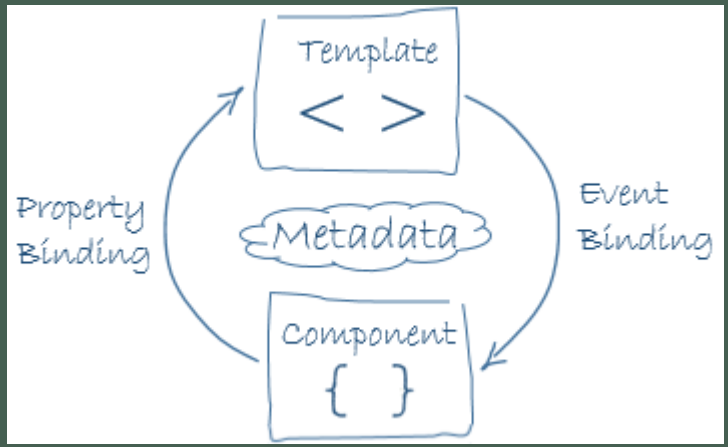
- ภายใน Component เราจะกำหนด Template ในรูปแบบของ HTML เพื่อให้ Angular Render เพจ
  - ใช้ HTML Tag ปกติเช่น <h2>, <p>
  - ใช้ Angular template-syntax เช่น *ngFor, {{hero.name}}, (click), [hero], <app-hero-detail>
- Directive คือส่วนที่กำหนดคำสั่งเพื่อควบคุมการแสดงผล
  - Structural Directive เปลี่ยน Layout โดยการเพิ่ม ลด หรือแทนที่ Element ใน DOM เช่น *ngFor, *ngIf
  - Attribute Directive เปลี่ยนลักษณะหรือพฤติกรรมของ Element ที่มีอยู่ (ส่วนมากใน <input>) เช่น [(ngModel)]



Data Binding

- Data Binding คือการเชื่อมโยงข้อมูลระหว่าง DOM และ Component สามารถมเชื่อมโยงระหว่าง Parent Component และ Child Component ได้
  - Interpolation ใช้เชื่อมโยง Property เช่น {{hero.name}}
  - Property Binding เหมือน Interpolation แต่เพิ่มการเชื่อมโยง Object จาก Parent ไปยัง Child เช่น [hero]
  - Event Binding ใช้เชื่อมโยงฟังก์ชันกับการกระทำเช่น (click)
- Two-way Data Binding คือการรวมเอา Property Binding และ Event Binding มารวมกัน เช่น [(NgModel)]





Pipe

- Pipe เป็นส่วนที่ใช้เปลี่ยนรูปแบบการแสดงผลของค่าต่าง ๆ

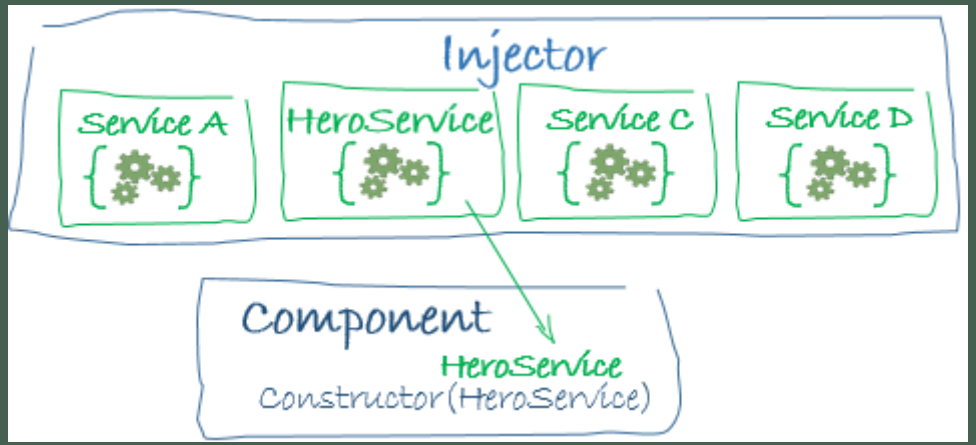
{{interpolated_value | pipe_name}}

- โดยส่วนมากแล้วเราจะใช้ Pipe กับค่าตัวเลขที่มีการแสดงผลแบบสกุลเงิน (currency) หรือวันที่ (date)

```
<!-- fullDate format: output 'Monday, June 15, 2015'-->
<p>The date is {{today | date:'fullDate'}}</p>
<!-- shortTime format: output '9:43 AM'-->
<p>The time is {{today | date:'shortTime'}}</p>
```

Service

- Service คือบริการที่สร้างขึ้นเพื่อให้แอพนำไปใช้งานได้ ประกอบไปด้วย ตัวแปร ฟังก์ชัน และ ฟีเจอร์ Service มักจะเป็นคลาสขนาดเล็ก ทำหน้าที่เฉพาะทาง เช่น เชื่อมต่อกับฐานข้อมูล หรือตรวจสอบข้อมูลเข้าจากผู้ใช้
- Service ต้องประกาศ @Injectable เผื่อสามารถนำไปใช้ใน Component ใดก็ได้

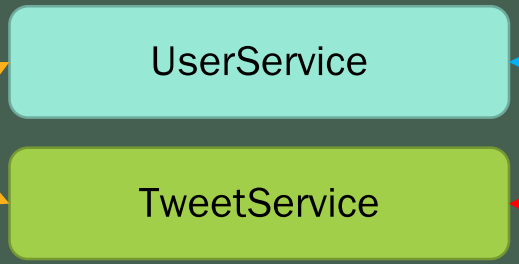


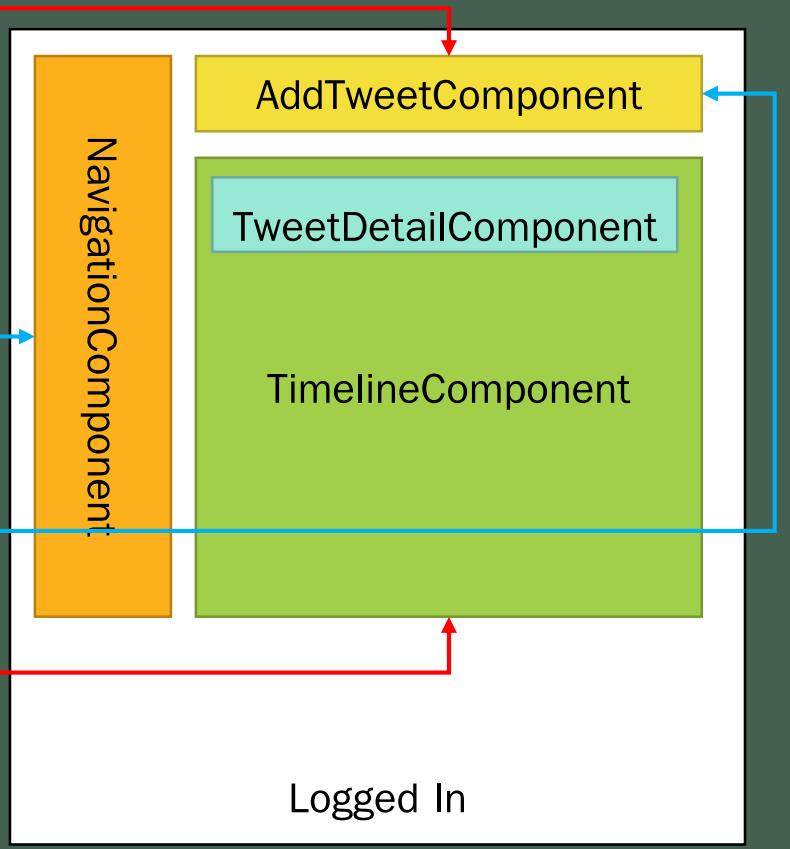
Routing

- Router เป็น NgModule ที่ให้บริการ Service สำหรับกำหนดเส้นทางไปยังหน้าต่าง ๆ ของแอพ รูปแบบการกำหนดเส้นทางใช้วิธีคล้ายกับการกระทำใน Browser
- Router โยง URL-like path ให้กับ View แทนที่จะเป็น Page ทำให้เมื่อผู้ใช้คลิก Link จะมี View ใหม่ที่ซ่อนอยู่มาแสดงผลแทนที่จะโหลดเพจใหม่ทั้งหมด
- ถ้าแอพเรามีขนาดใหญ่ ต้องโหลดหลาย Module พร้อมกัน อาจจะทำให้การโหลดช้าลง ปัญญานี้สามารถแก้ไขได้ด้วยการใช้ Asynchronous Routing เพื่อโหลด Module on Demand

Project Checkpoint 2

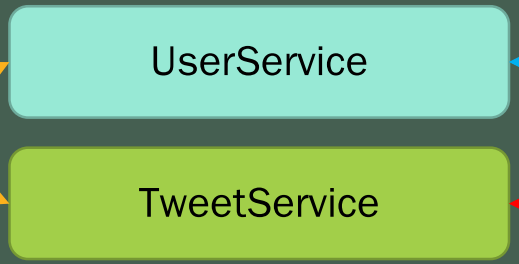
- ออกแบบ UI, Component, Service
- Save เป็นไฟล์รูปแล้วแปะไว้ที่หน้า Proposal

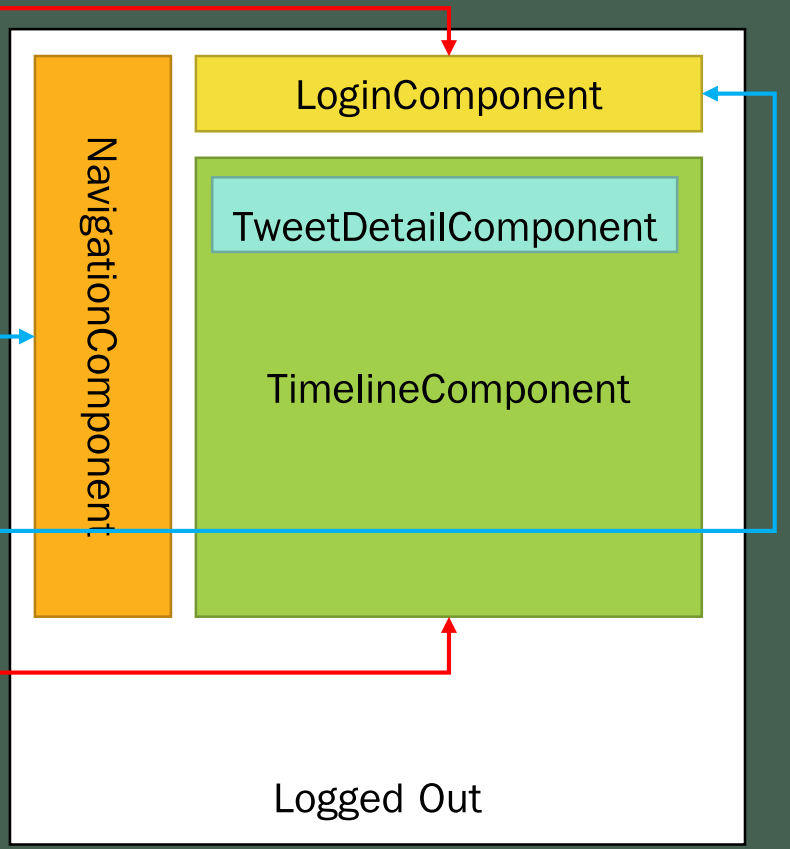




Project Checkpoint 2

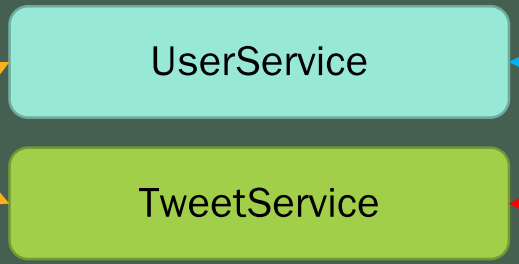
- ออกแบบ UI, Component, Service
- Save เป็นไฟล์รูปแล้วแปะไว้ที่หน้า Proposal

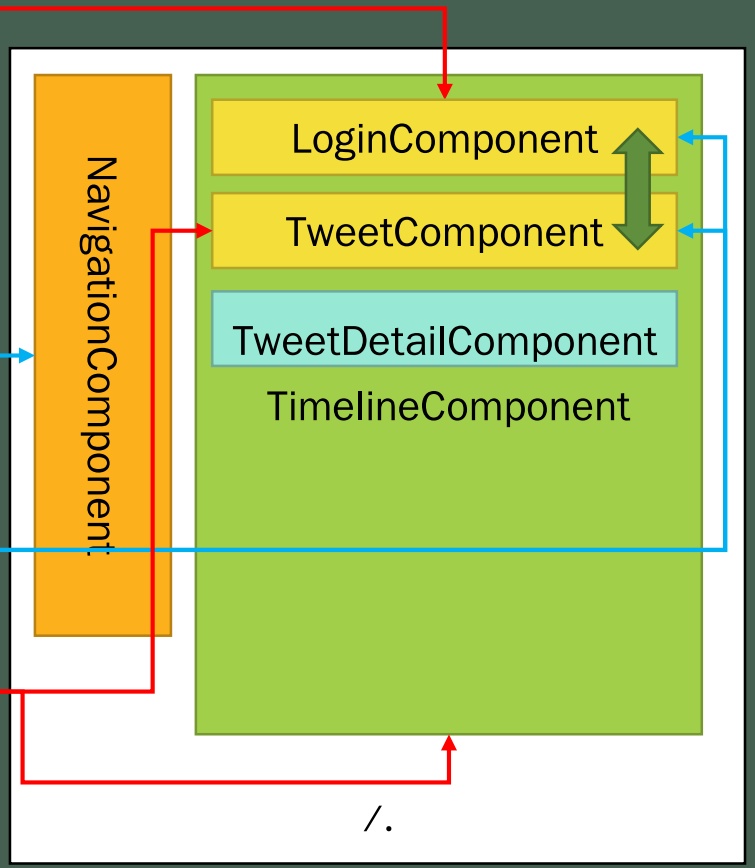




Project Checkpoint 2

- ออกแบบ UI, Component, Service
- Save เป็นไฟล์รูปแล้วแปะไว้ที่หน้า Proposal





Project Checkpoint 2

- ออกแบบ UI, Component, Service
- Save เป็นไฟล์รูปแล้วแปะไว้ที่หน้า Proposal

UserService

TweetService

NavigationComponent

ProfileComponent

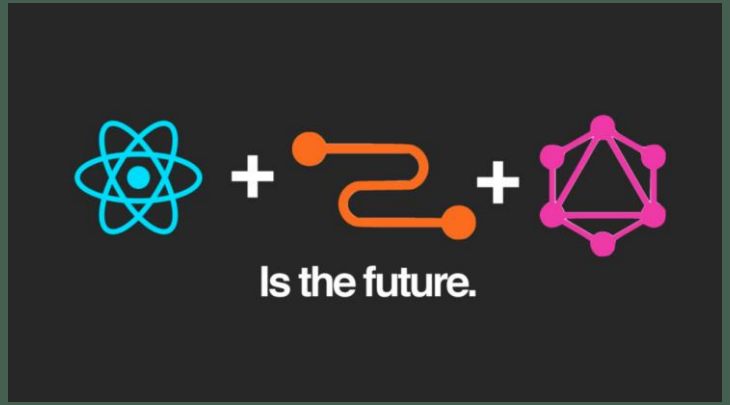
/profile

Beyond Web Application

- ในระหว่างที่เราเรียนรู้การสร้างเว็บแอพผ่าน Framework ต่าง เราจะพบว่าแต่ละ Framework มีข้อดีข้อเสียที่แตกต่างกัน แต่ในฐานะ Develop เรามักจะยอมรับกับข้อจำกัดนั้น
- แต่สำหรับบริษัทใหญ่ ๆ ที่ต้องการพัฒนาเว็บแอพของตัวเองให้เกินข้อจำกัดของ Framework เหล่านั้น การนำ Framework มาพัฒนาต่อเองจึงเป็นเรื่องสำคัญ
- การพัฒนา Framework นั้นทำได้สองแบบ คือ Branch-off และ Add-ons
- ในคลาสนี้เราจะยกตัวอย่างการพัฒนา Framework แบบ Add-ons ของ Facebook
- https://www.youtube.com/watch?v=WxPtYJRjLLO

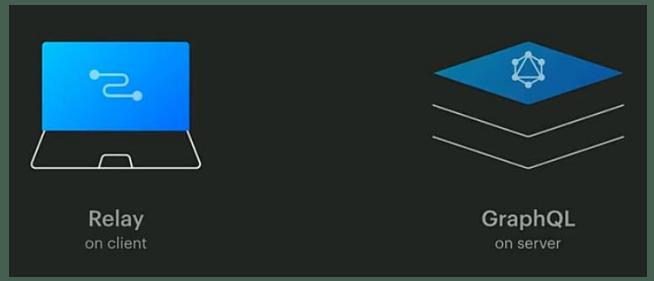
New Facebook

- Facebook ต้องการเอาจุดเด่นของ Mobile App ในด้านความง่ายและความเร็ว มาไว้กับ Web App ซึ่งการตัดสินใจนี้ทำให้ทีม Facebook ต้องคิดค้นโครงสร้างใหม่ของเว็บแอพ
- โดยโครงสร้างใหม่ถูกสร้างขึ้นด้วย React, GraphQL และ Relay (ทั้งหมดถูกพัฒนาโดย Facebook) ซึ่ง Relay เอามาแทน Redux ที่เคยใช้กันอยู่

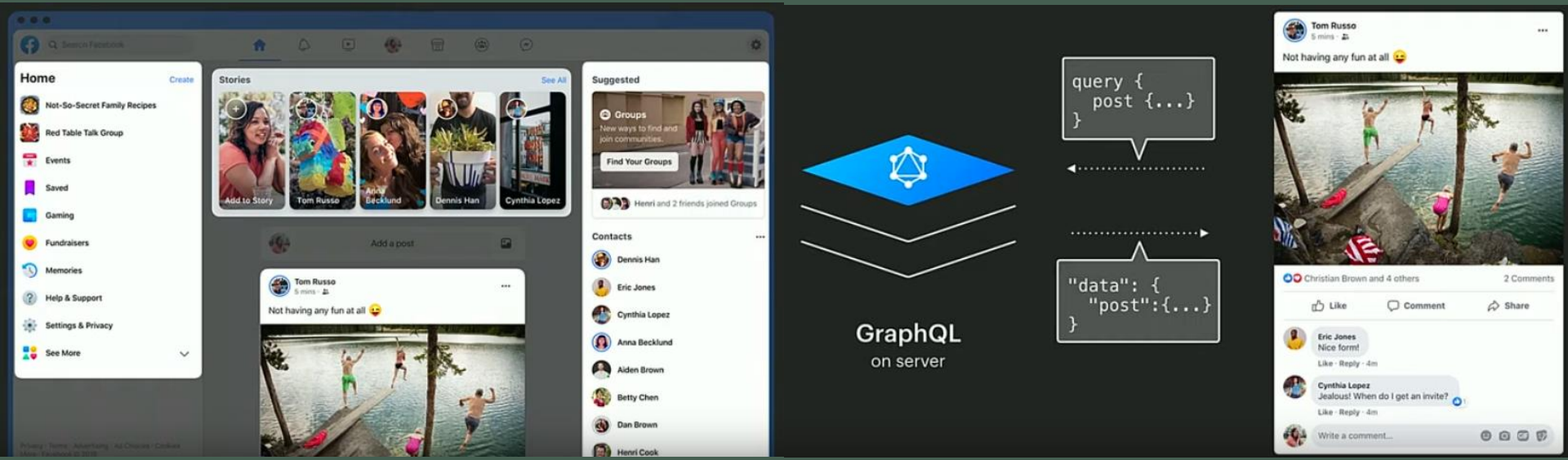


GraphQL and Relay

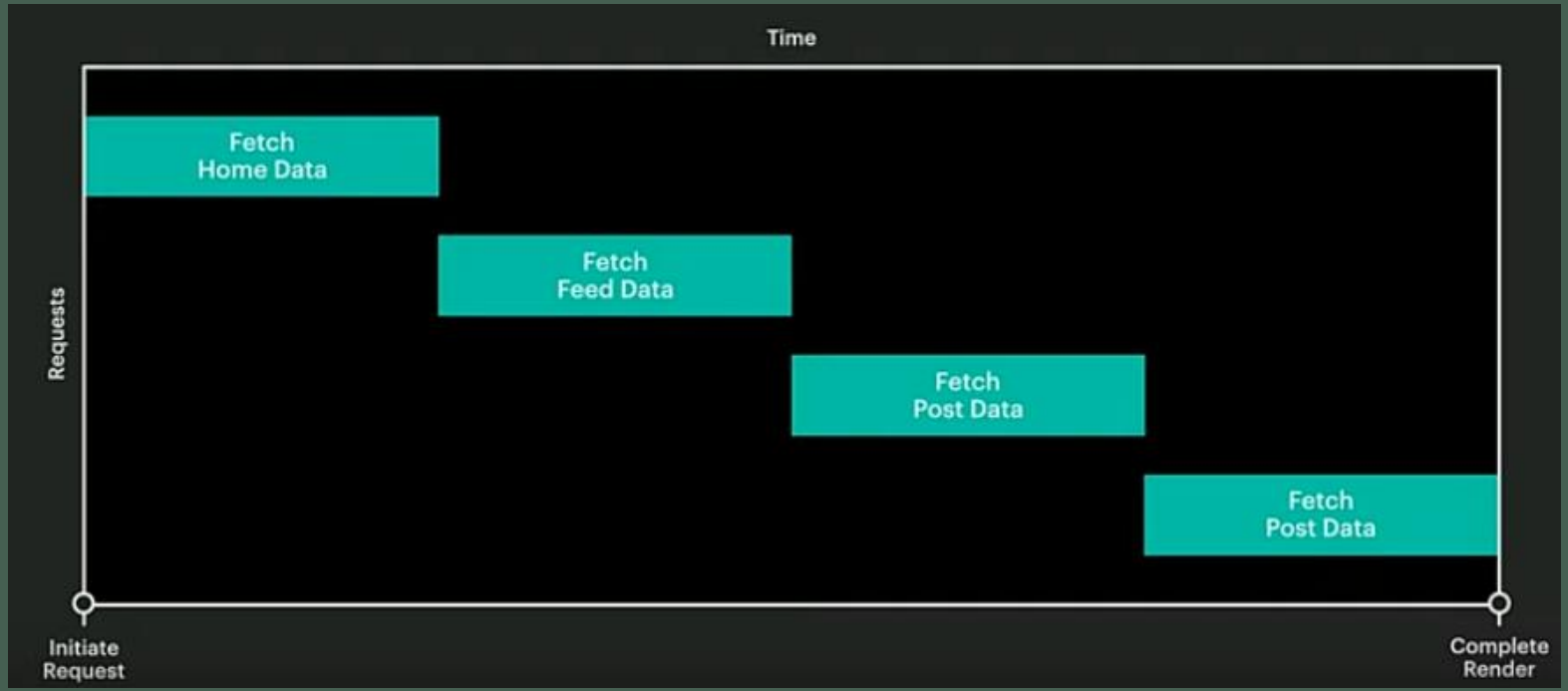
- GraphQL เป็นภาษาที่ใช้การ Query ข้อมูลจากเซิฟเวอร์ โดยโครงสร้างของข้อมูลจะถูก Client เป็นตัวกำหนด ข้อมูลจากแหล่งข้อมูลเดียวกันอาจจะใช้จำนวน Attribute ในการแสดงผลไม่เท่ากัน ฉะนั้นแทนที่เราจะพัฒนา API เพื่อส่งข้อมูลหลายโครงสร้าง เราจะพัฒนา API ที่รับโครงสร้างมาแล้วคืนข้อมูลตามโครงสร้างนั้น
- Relay เป็น JavaScript Framework ที่ใช้ดึงข้อมูลในแอพที่พัฒนาบน React
- Facebook เลือกใช้ GraphQL บน Server แล้วใช้ Relay พัฒนา Client



Single Feed Fetching

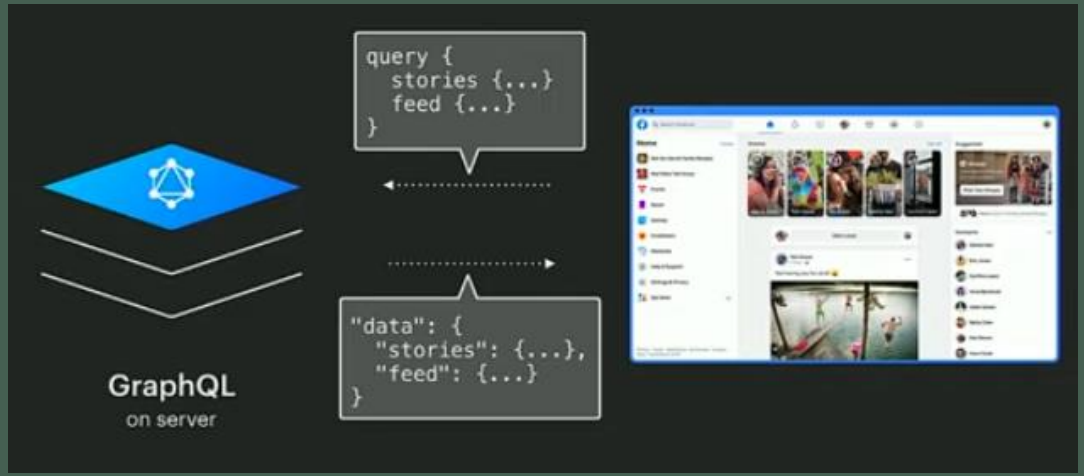


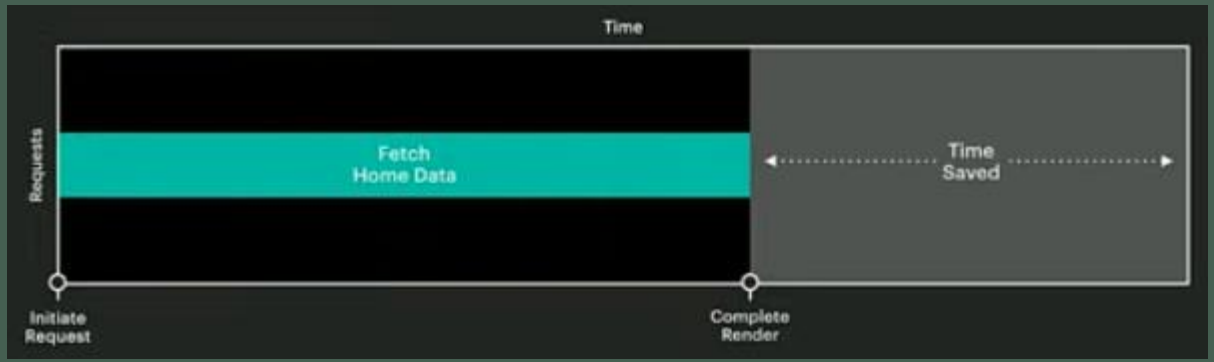
Request Waterfall



Solved with GraphQL

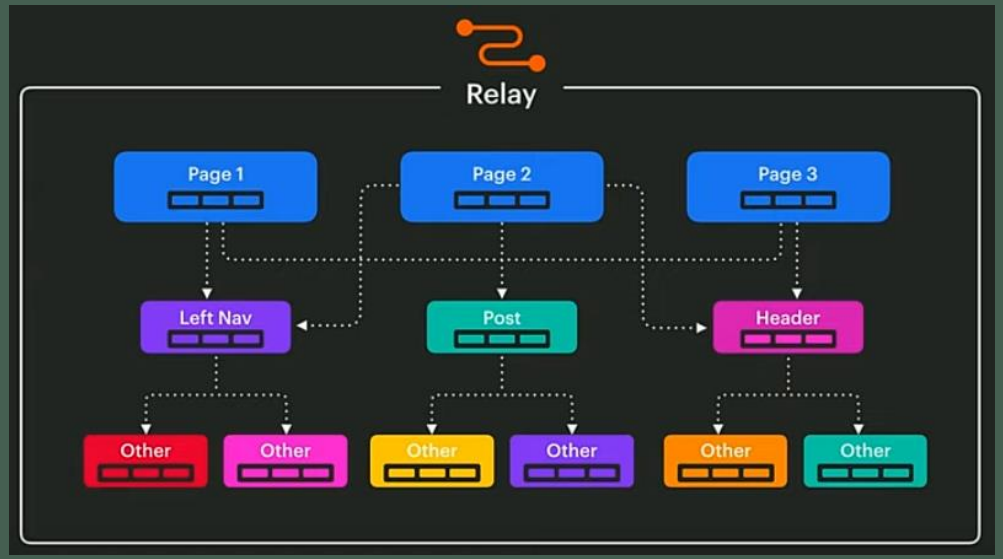
- ส่ง Top-Level Query เพื่อลดเวลาในการเชื่อมต่อกับเซิฟเวอร์
- แต่ยังมีปัญหาเรื่องการเปลี่ยนแปลงข้อมูลที่ต้องไปอัพเดต Query ของ





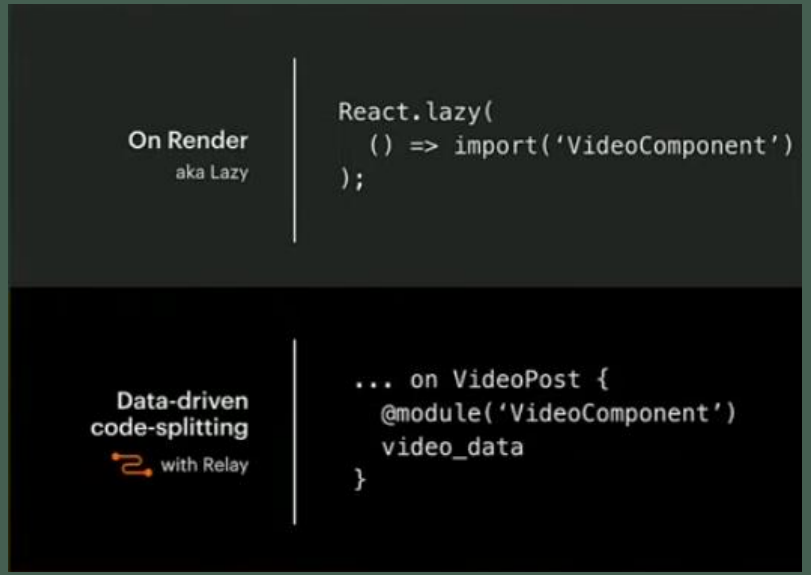
Solved by Relay

- Relay สามารถ Track ข้อมูลที่ต้องการของแค่ละเพจได้ โดยเราไม่ต้องเขียน Code เพื่อส่งข้อมูลจาก Component ไปยัง Page ที่ต้องการ
  - React ส่งข้อมูลทางเดียวแต่ Angular ส่งข้อมูลสองทาง

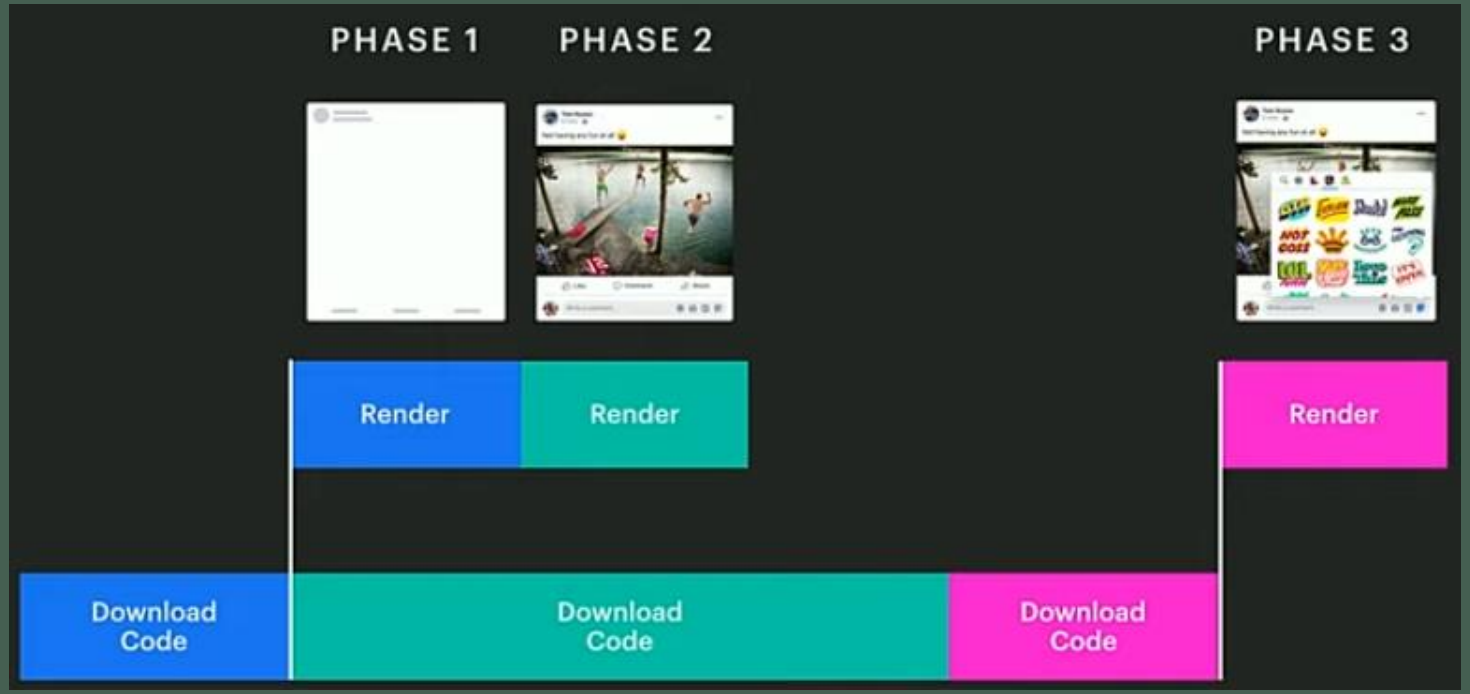


Code Size and Efficiency

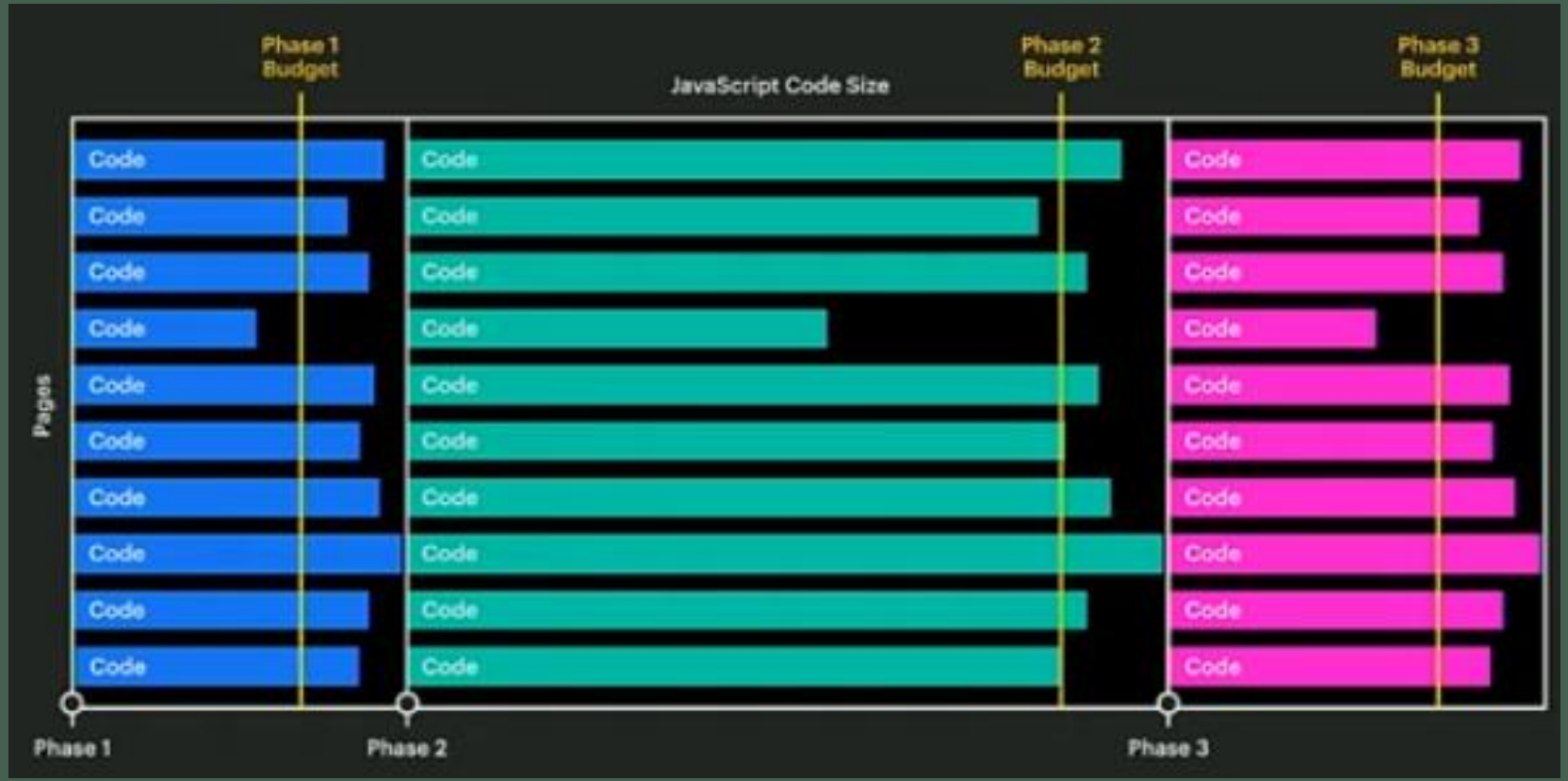
- อีกหนึ่งปัญหาที่ Facebook ต้องเผชิญคือรูปแบบของข้อมูลและการแสดงผลข้อมูลนั้น แต่ละโพสอาจจะเป็นแค่ข้อความ รูปภาพ เสียง หรือ วีดีโอ ซึ่งแต่ละแบบต้องใช้ Code ในการแสดงผลต่างกัน
- ถ้าเราโหลด Code ทุกแบบมาตั้งแต่ตอนแรกก็จะทำให้เสียเวลามาก แต่ถ้าเราโหลดข้อมูลมาก่อนแล้วค่อยโหลด Code ที่ใช้แสดงผลข้อมูลนั้นก็ต้องส่ง Request ไปยัง Server สองรอบ
- ทั้งสองปัญหาสามารถรวมกันเป็นปัญหาเดียวคือ เราไม่รู้ว่าเราต้องการ Code อะไรตอนส่ง Request ครั้งแรก
- วิธีการแก้ปัญหาคือการส่งข้อมูลพร้อม Code ที่ต้องใช้แสดงผลข้อมูลนั้นพร้อมกัน



Code Splitting with Phases



Limiting Code Size



CSS Code Size

- สำหรับ Facebook ขนาดของ Code ที่เกี่ยวข้องกับ CSS มีขนาดถึง 2.3MB (450KB Compressed) สำหรับหน้า Homepage อย่างเดียว
- CSS แบบเก่าเน้นการแยกการแสดงผลตามคลาสที่กำหนด ส่งผลให้ในหลาย ๆ คลาสอาจจะมีกฎเหมือนกัน

Atomic Stylesheet

- แต่ละกฎจะมีคลาสเป็นตัวของมันเอง แต่ละ Component สามารถเป็นได้หลายคลาส ถ้าเราสร้างเพจใหม่เราก็สามารถใช้กฎเดิมซ้ำได้โดยไม่ต้องประกาศคลาสมาใหม่

```
<Component1 classNames=".classA .classC .classD"/>

<Component2 classNames=".classA .classB .classD .classE"/>
```

```
.classA { cursor: default; }
.classB { background-color: var(--fds-active-icon); }
.classC { background-color: var(--fds-gray-25); }
.classD { margin-left: 0px; }
.classE { justify-self: flex-start; }
```