



Vue.js (3.x)

สำหรับผู้เริ่มต้น




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



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



พื้นฐานก่อนลุย Vue.js








สอน Vue.js Basic to Advance
(ทยอยลงคลิป)

วิดีโอ 20 รายการ • การดู 106,591 ครั้ง • อัปเดตล่าสุด
เมื่อ 25 เม.ย. 2018

≡+ ↺ ↻ ...

 Kong Ruksiam ติดตามแล้ว 

- 1  สอน Vue.js ตอนที่ 1 - Hello World & Expression
Kong Ruksiam 11:34
- 2  สอน Vue.js ตอนที่ 2- Data & Method
Kong Ruksiam 4:40
- 3  สอน Vue.js ตอนที่ 3 - Data Binding
Kong Ruksiam 7:27
- 4  สอน Vue.js ตอนที่ 4 - Event Handler
Kong Ruksiam 6:52
- 5  สอน Vue.js ตอนที่ 5 - KeyBoard Event
Kong Ruksiam 5:19



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



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

ผู้เรียนต้องมีพื้นฐานอะไรบ้าง

- HTML5 เบื้องต้น
- CSS3 เบื้องต้น
- JavaScript เบื้องต้น
- JavaScript ES6
- Bootstrap 5 (Optional)



เรียนปูพื้นฐานได้ที่ไหน



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



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

เนื้อหา HTML , CSS , JavaScript

ปูพื้นฐาน HTML | CSS3 | JavaScript | Bootstrap5 ▾



ปูพื้นฐานการสร้างเว็บด้วย HTML5 [2020]

KongRuksiam Official
ดูทั้งเพลย์ลิสต์



ออกแบบเว็บไซต์ด้วย CSS3 [2020]

KongRuksiam Official
ดูทั้งเพลย์ลิสต์



ปูพื้นฐานการสร้างเว็บด้วย JavaScript เบื้องต้น [2020]

KongRuksiam Official
อัปเดตแล้ววันนี้
ดูทั้งเพลย์ลิสต์



สอน Bootstrap 5 สำหรับผู้เริ่มต้น [2021]

KongRuksiam Official
ดูทั้งเพลย์ลิสต์



เนื้อหา JavaScript Workshop [Promote Version]

KongRuksiam Official
ดูทั้งเพลย์ลิสต์



ปูพื้นฐาน JavaScript ES6 สำหรับมือใหม่

KongRuksiam Official



คอร์ส JavaScript 30 Workshop | ฉบับ 2020 [Family]

KongRuksiam Official



คอร์ส JavaScript 25 Workshop (For Supporter)

KongRuksiam Official



สแกนเพื่อเข้าเรียน

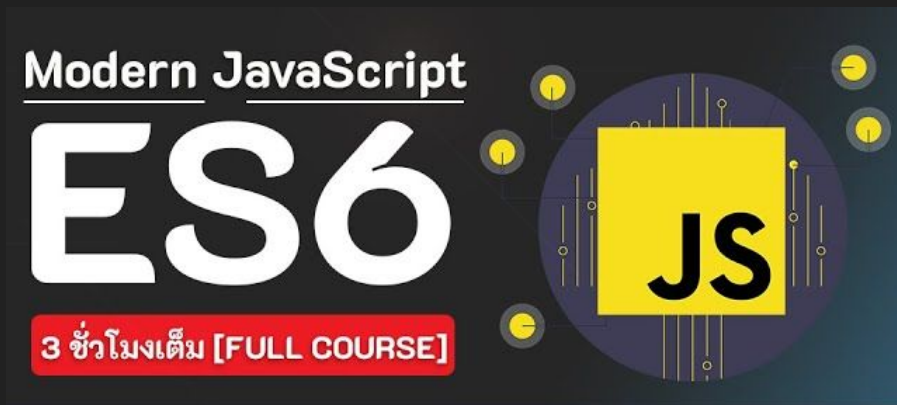


<https://www.youtube.com/c/KongRuksiamOfficial/>



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

JavaScript ES6 (สำคัญมาก!!!!!!)



สแกนเพื่อเข้าเรียน

รู้จักกับ Vue.js



Vue.js คือ Frontend Framework

ที่ใช้สำหรับสร้าง User Interface (UI) หรือ
หน้าจอของแอปพลิเคชันในรูปแบบไดนามิก
ที่สามารถอัปเดตบางส่วนของแอปพลิเคชัน
โดยไม่ต้องโหลดแอปพลิเคชันขึ้นมาใหม่
ทั้งหมด



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

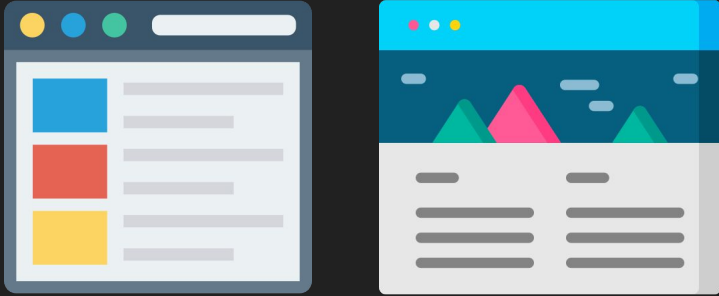


<https://www.facebook.com/KongRuksiamTutorial/>



- Frontend ??
- Framework ??

Frontend VS Backend



Frontend คือ การพัฒนาโปรแกรมระบบหน้าบ้าน (UI : User Interface หรือ หน้าตาของแอปพลิเคชัน) โดยผู้ใช้งานสามารถมองเห็นและมีส่วนร่วมหรือโต้ตอบภายใน Web Browser ได้



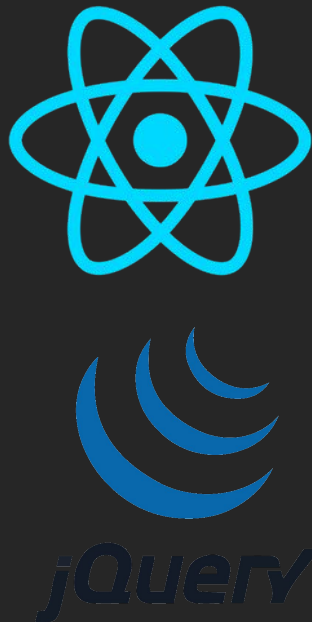
Backend คือ การพัฒนาโปรแกรมหลังบ้านหรือการทำงานเบื้องหลังในแอป เช่น การทำงานกับฐานข้อมูล เป็นต้น โดยผู้ใช้งานไม่สามารถมีส่วนร่วมหรือโต้ตอบได้

Framework vs Library

Framework



Library



Framework vs Library

Framework

ข้อดี

- มีรูปแบบที่ชัดเจนเป็นแบบแผนเหมาะสมกับงานที่ทำเป็นทีมต้องเขียนตามรูปแบบที่กำหนดไว้
- มีเครื่องมือทุกอย่างพร้อมให้เราใช้งานได้เลย

ข้อเสีย

- ไม่มีความยืดหยุ่น
- ปรับปรุงแก้ไขการทำงานได้ยาก

Library

ข้อดี

- เลือกเครื่องมือหรือสิ่งที่ต้องการมาใช้งานในระบบของเราได้เลย
- มีความยืดหยุ่นสูง

ข้อเสีย

- ต้องทำทุกอย่างด้วยตนเอง

แนวคิดของ Vue.js

Vue.js หรือเรียกสั้นๆว่า Vue ถูกนำมาใช้สร้างหน้าเว็บ โดยที่สามารถแบ่งการแสดงผลหน้าเว็บออกเป็นส่วนย่อยหลายๆส่วนได้ โดยที่ไม่ต้องเขียนหน้าเว็บเก็บไว้ในไฟล์เดียว เพื่อให้ง่ายต่อการจัดการ แล้วค่อยนำส่วนย่อยดังกล่าวมาประกอบกันในภายหลัง เราจะเรียกองค์ประกอบที่แบ่งออกเป็นส่วนย่อย ๆ นี้ว่า

“Component”

ข้อดีของการสร้าง Component คือ สามารถที่จะออกแบบแล้วนำกลับมาใช้งานในภายหลังได้ โดยที่ไม่ต้องเสียเวลาเขียนใหม่

แนวคิดของ Vue.js



การสร้างหน้าเว็บแบบเดิม ต้องเขียนโค้ดทั้งหมดไว้ในไฟล์เดียว ซึ่งส่งผลให้หน้าเว็บทำงานช้ามาก เพราะต้องโหลดเนื้อหาใหม่ทั้งหมดในกรณีที่มีการเปลี่ยนแปลงการทำงานในหน้าเว็บ

<https://publicwebsite-ec540.web.app/>

แนวคิดของ Vue.js



แนวคิดของ Vue.js



แนวคิดของ Vue.js

Profile
Component



Kong Ruksiamza

เขียนไปเรื่อย เขียนก็ไม่เขียน


KONG RUKSIAMZA FOLLOWS

 Krissanawat Kaewsanmu

 Chanon Yakhai

 Ploy Thanasornsawan

 nondev studio;

 Anan Yoothongdee

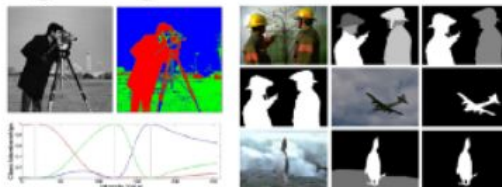
See all (177)

Follows
Component

May 4, 2020

แจก Slide ทฤษฎีประกอบเนื้อหาการประมวล ภาพดิจิทัล (Digital Image Processing)

Image Segmentation



การแยกบริเวณของภาพ ซึ่งการแยกบริเวณนั้นจะทำให้ภาพที่เป็นวัตถุที่สนใจออกจากพื้นหลัง ที่จะทำให้ทราบว่าในภาพมีวัตถุอยู่กี่ชิ้นและพิกเซลใดเป็นของวัตถุชิ้นใด ซึ่งกระบวนการดังกล่าวถือเป็นพื้นฐานของการประมวลผลขั้นสูงที่จะนำไปสู่การตัดสินใจเกี่ยวกับคุณภาพของผลิตภัณฑ์ต่อไปและจะพบว่าวิธีการแยกบริเวณนั้นจะแบ่งออกเป็น 2 ประเภทหลักๆ ด้วยกัน

Read more · 1 min read



Content Component

Navbar Component

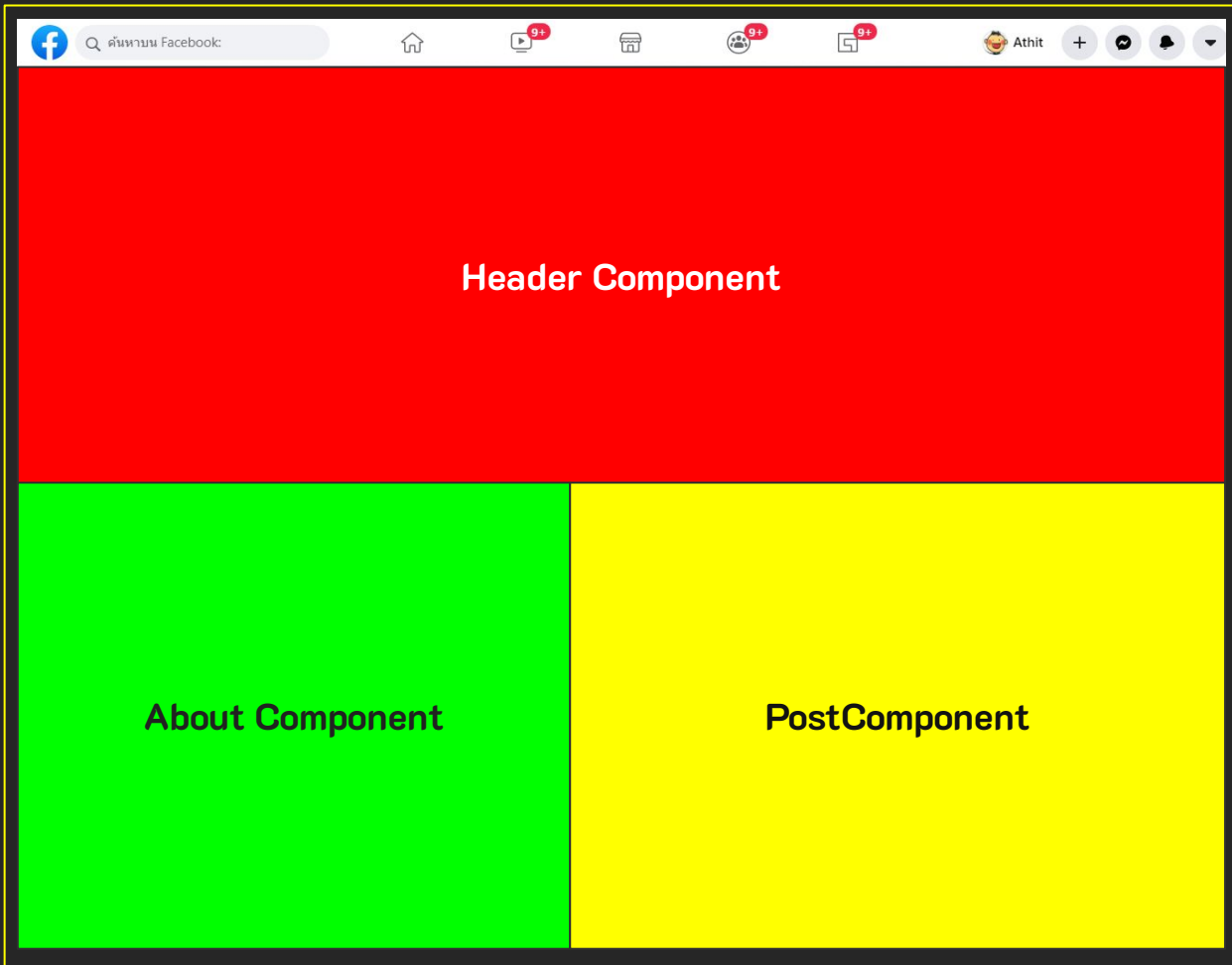


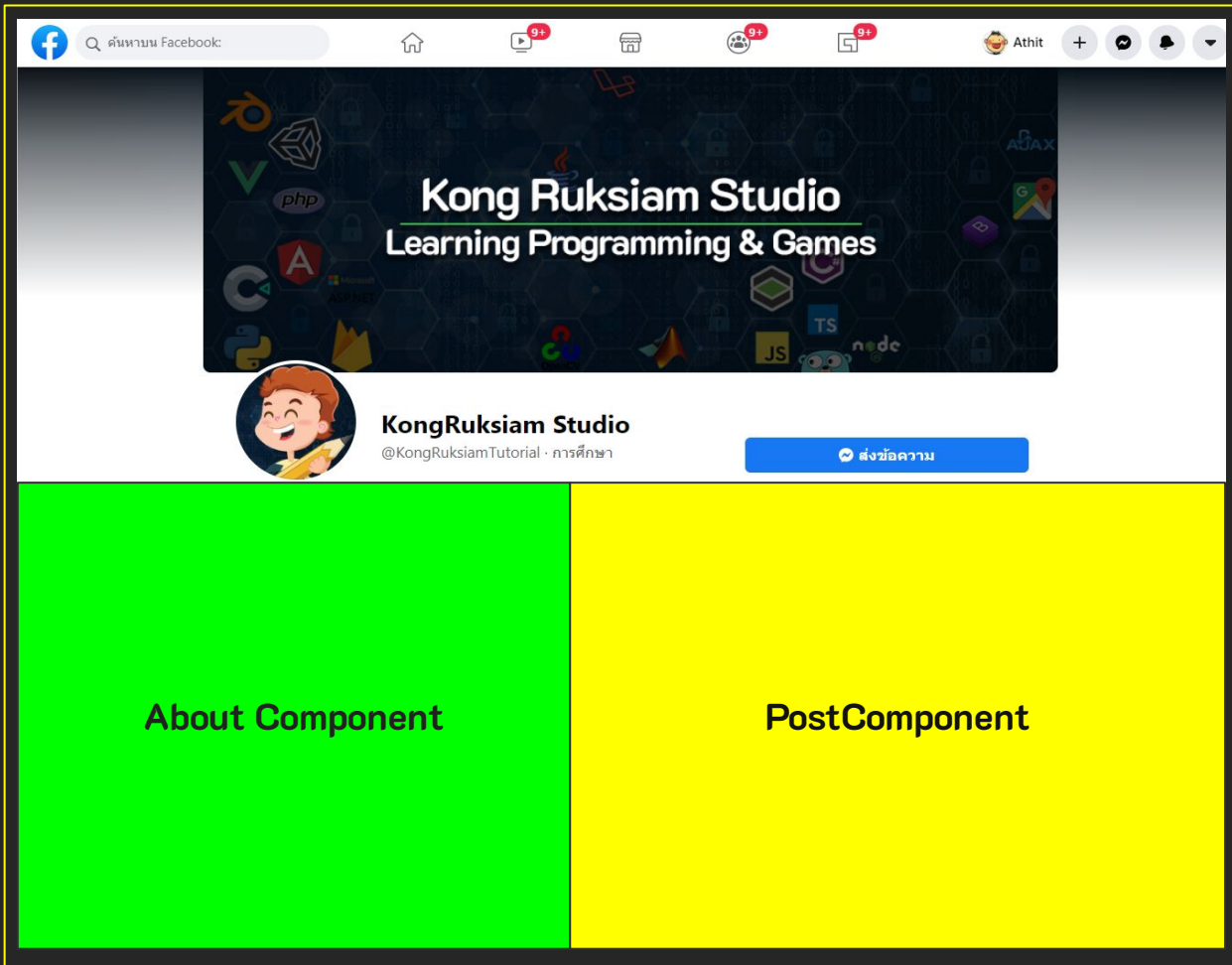
```
graph TD; Navbar[Navbar Component] --- Header[Header Component]; Header --- About[About Component]; Header --- Post[PostComponent];
```

Header Component

About Component

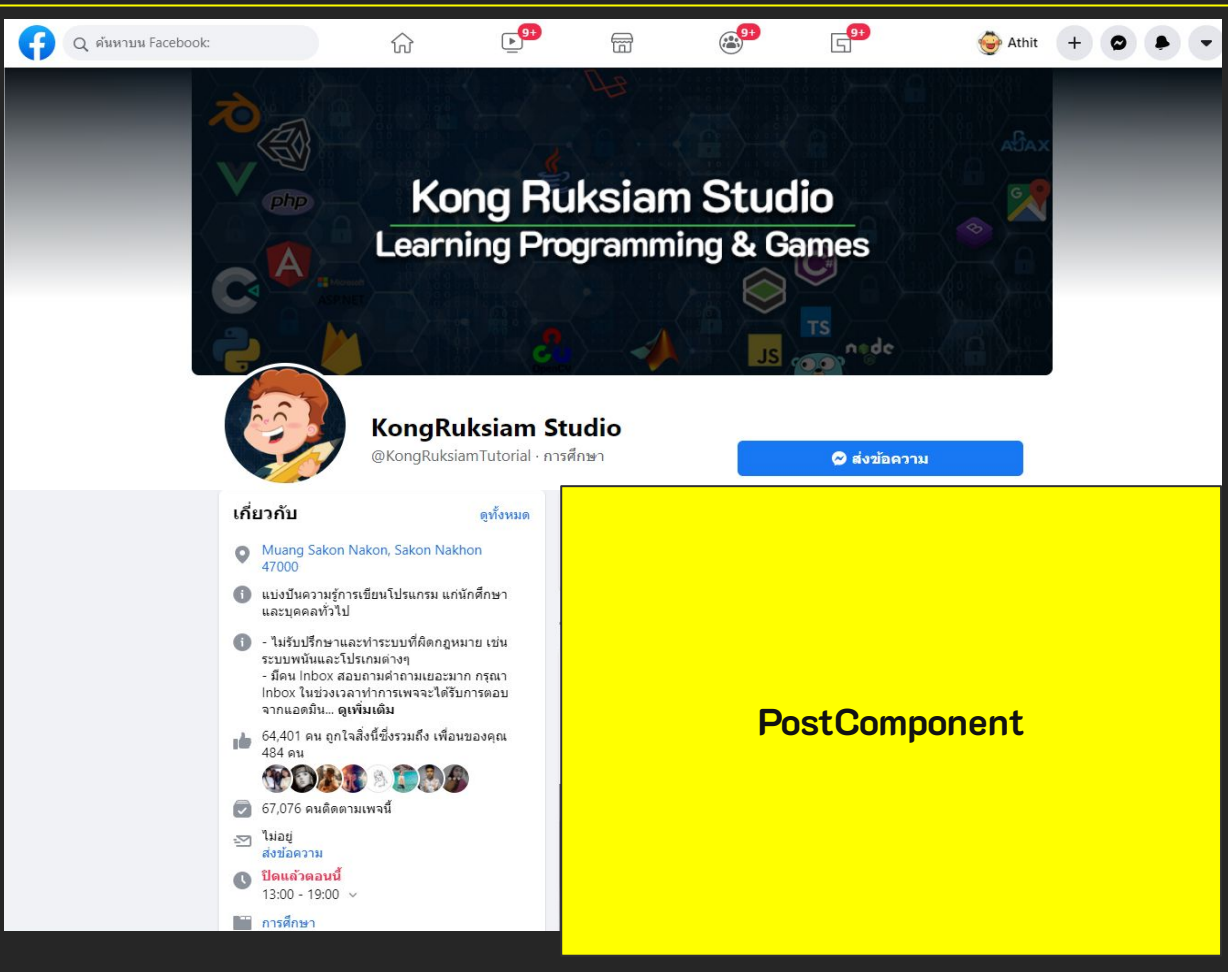
PostComponent





About Component

PostComponent



- HTML , CSS , JS



จุดเด่นของ Vue.js

- มีขนาดเล็ก ทำงานได้อย่างรวดเร็ว
- มีโครงสร้างการเขียนที่ไม่ซับซ้อน
- แบ่งแอปพลิเคชันออกเป็นส่วนย่อยๆ หรือเรียกว่า Component ง่ายต่อการปรับปรุงแก้ไขและพัฒนาระบบ
- เป็นรูปแบบ Dynamic Webpage เนื้อหาในหน้าเว็บมีการเปลี่ยนแปลงตามเงื่อนไขที่กำหนด

จุดเด่นของ Vue.js

- มีเครื่องมือและ plugin ต่างๆให้เลือกใช้งานมากมาย
- ทำงานข้าม Platform ได้ (Window , Mac , Linux)
- รองรับการทำงานกับ Browser ได้ทุกตัว เช่น Microsoft Edge , Google Chrome , Safari , Firefox , Opera เป็นต้น

Imperative & Declarative

Imperative Programming

เป็นรูปแบบการเขียนโปรแกรมแบบเดิมที่ใช้ใน JavaScript ในการบอกว่าเว็บของเราต้องการอยากรจะสร้างอะไร แล้วให้กระทำอะไร เช่น

```
const btnEl = document.createElement('button')
```

```
btnEl.innerHTML = 'Send Data'
```

```
document.body.appendChild(btn)
```

Declarative Programming

เป็นรูปแบบการเขียนโปรแกรมที่ได้รับความนิยมในปัจจุบันเนื่องจากทำให้
โค้ดมีความกระชับอ่านแล้วเข้าใจง่าย มีความหมายที่ตรงตัว

```
<Button>Send Data</Button>
```

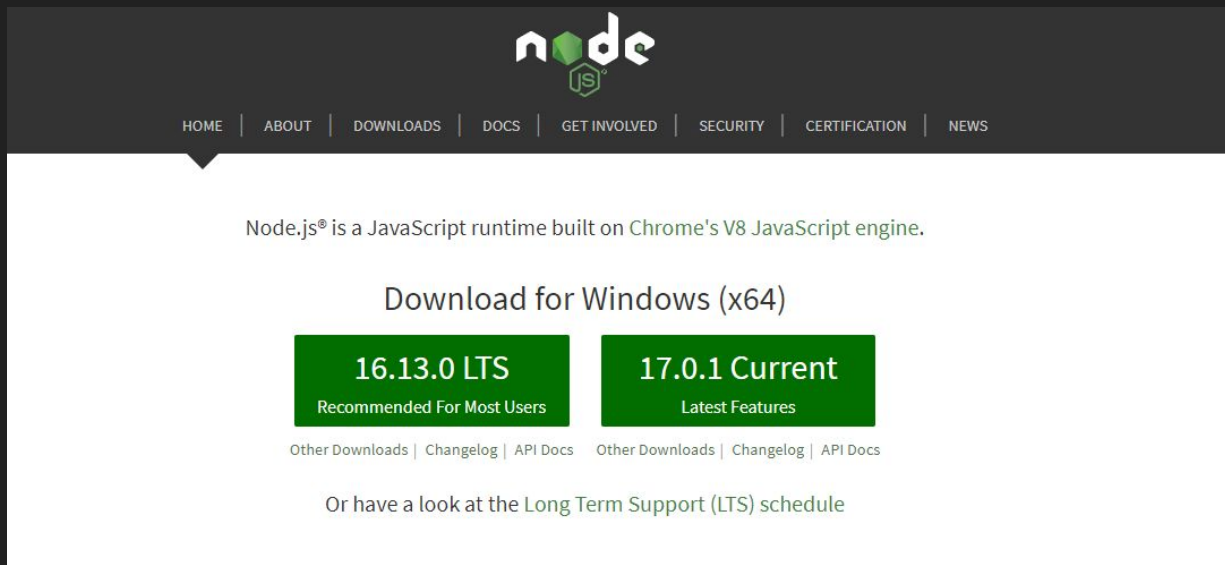


เตรียมเครื่องมือให้พร้อม

เครื่องมือพื้นฐาน

- Node.js
- Visual Studio Code + Extensions
- Vue.js Devtools (Google Chrome)

1. ติดตั้ง Node.js



2. ติดตั้ง Visual Studio Code



Visual Studio Code

2.1 ติดตั้ง Extensions

- Vetur
- AutoRename Tag
- Color Highlight
- Prettier Code Formatter
- Live Server (Optional)

3 Chrome Extensions

- **Vue.js devtools**

<https://v3.vuejs.org/guide/installation.html#vue-devtools>



ติดตั้ง Vue CLI (Command Line Interface)

หน้าที่ของ Vue CLI

- สร้าง Vue แอปพลิเคชันด้วยคำสั่งบรรทัดเดียว
- ติดตั้งเครื่องมือหรือ Plugin เสริมเข้าไปในโปรเจกต์ได้ง่ายกว่า CDN
- ทดสอบรันแอปพลิเคชันโดยจำลองการทำงานของ Server ที่เครื่องของผู้พัฒนา (localhost)

ติดตั้ง Vue CLI (Command Line Interface)

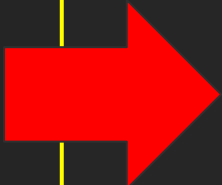
การติดตั้ง @vue/cli จะมาพร้อมกับเครื่องมือที่จำเป็นสำหรับพัฒนาเว็บแอปพลิเคชัน ประกอบด้วย

1. **Babel** เป็น JavaScript Compiler ที่ทำให้สามารถเขียน JavaScript เวอร์ชันใหม่ๆ ใน Vue ได้ เช่น ES6 หรือ ES7
2. **ESLint** ตรวจสอบไวยากรณ์ JavaScript ว่าเขียนถูกต้องหรือไม่
3. **webpack** สำหรับจัดการและแปลงไฟล์ประเภทต่างๆ ให้อยู่ในรูปแบบที่ Browser อ่านได้ เช่น ไฟล์รูปภาพ เป็นต้น

สร้างโปรเจค Vue

1. ติดตั้ง @vue/cli
2. vue create ชื่อโปรเจค
3. npm run serve

โครงสร้างโปรเจค



```

▼ my-app
  > node_modules
  ▼ public
    ★ favicon.ico
    <> index.html
  ▼ src
    ▼ assets
      🖼 logo.png
    ▼ components
      ▼ HelloWorld.vue
      ▼ App.vue
    JS main.js
    💎 .gitignore
    📄 babel.config.js
    {} package-lock.json
    {} package.json
    ⓘ README.md

```

- **node_modules** - เก็บโมดูลหรือไลบรารีที่ทำงานภายในโปรเจคของเรา
- **public** - เก็บไฟล์ public ต่างๆ และ index.html
- **src** - สำหรับเก็บ component หรือโครงสร้างของแอปพลิเคชัน และ asset ที่ใช้งาน
- **package.json** - เก็บข้อมูลต่างๆรวมถึง package ที่จะใช้ทำงานภายในโปรเจค

ควบคุมการทำงานของแอปพลิเคชัน (src)

```
✓ src
  ✓ assets
    logo.png
  ✓ components
    ▼ HelloWorld.vue
    ▼ App.vue
  JS main.js
```



โครงสร้างโปรเจค

สำหรับควบคุมการทำงานของแอปพลิเคชันหลักๆจะอยู่ในโฟลเดอร์ src ซึ่งประกอบด้วยไฟล์และโฟลเดอร์ดังนี้

- src/assets โฟลเดอร์สำหรับจัดการเกี่ยวกับภาพหรือไฟล์มัลติมีเดีย
- src/components ใช้เก็บส่วนประกอบย่อย (component) ในแอปพลิเคชัน
- src/App.vue เป็นไฟล์ Component หลักที่อยู่ในแอปพลิเคชัน (root component)
- src/main.js เป็นไฟล์ script หลัก ที่ควบคุมการทำงานของแอปพลิเคชันโดยเชื่อมการทำงานระหว่าง App.vue และ Index.html

โครงสร้างโปรเจค

```
└─ public
  └─ ★ favicon.ico
  └─ <> index.html
└─ src
  └─ assets
    └─ 🖼 logo.png
  └─ components
    └─ ▼ HelloWorld.vue
    └─ ▼ App.vue
  └─ JS main.js
```

- ไฟล์ index.html ที่อยู่ในโฟลเดอร์ public จะเป็นไฟล์ HTML สำหรับแสดงผลลัพธ์ใน Browser ซึ่งเนื้อหาที่นำมาแสดงผลนั้นมาจาก Components

โครงสร้างโปรเจค

สิ่งที่เราสนใจใน index.html ก็คือ คำสั่งที่อยู่ใน <body> ตรงส่วนของ
<div id="app"></div> โดยจะนำเอาเนื้อหาที่อยู่ใน App.vue มาแสดงผลในพื้นที่ดังกล่าว

```
index.html x
my-app > public > index.html > ...
1 <!DOCTYPE html>
2 <html lang="">
3 <head>
4   <meta charset="utf-8">
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <meta name="viewport" content="width=device-width,initial-scale=1.0">
7   <link rel="icon" href="%= BASE URL %>favicon.ico">
8   <title><%= htmlWebpackPlugin.options.title %></title>
9 </head>
10 <body>
11   <noscript>
12     <strong>We're sorry but <%= htmlWebpackPlugin.options.title %> doesn't
13   </noscript>
14   <div id="app"></div>
15   <!-- built files will be auto injected -->
16 </body>
17 </html>
```

index.html

```
JS main.js x
my-app > src > JS main.js
1 import { createApp } from 'vue'
2 import App from './App.vue'
3
4 createApp(App).mount('#app')
5
```

main.js เชื่อม App Component
กับ index.html

โครงสร้างโปรเจค

สิ่งที่เราสนใจใน index.html ก็คือ คำสั่งที่อยู่ใน <body> ตรงส่วนของ
<div id="app"></div> โดยจะนำเอาเนื้อหาที่อยู่ใน App.vue มาแสดงผลในพื้นที่ดังกล่าว

```
index.html ×
my-app > public > index.html > ...
1 <!DOCTYPE html>
2 <html lang="">
3 <head>
4   <meta charset="utf-8">
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <meta name="viewport" content="width=device-width,initial-scale=1.0">
7   <link rel="icon" href="%= BASE URL %>favicon.ico">
8   <title><%= htmlWebpackPlugin.options.title %></title>
9 </head>
10 <body>
11   <noscript>
12     <strong>We're sorry but <%= htmlWebpackPlugin.options.title %> doesn't
13   </noscript>
14   <div id="app"></div>
15   <!-- built files will be auto injected -->
16 </body>
17 </html>
```

index.html

```
JS main.js ×
my-app > src > JS main.js
1 import { createApp } from 'vue'
2 import App from './App.vue'
3
4 createApp(App).mount('#app')
5
```

main.js เชื่อม App Component
กับ index.html

แผนภาพการทำงาน

```
<template>
  
  <h1>KongRuksiam Studio</h1>
</template>

<script>

export default {
  name: 'App',
}
</script>
```

App.vue

```
index.html X
my-app > public > index.html > ...
1  <!DOCTYPE html>
2  <html lang="">
3  <head>
4    <meta charset="utf-8">
5    <meta http-equiv="X-UA-Compatible" content="IE=edge">
6    <meta name="viewport" content="width=device-width,initial-scale=1.0">
7    <link rel="icon" href="<%= BASE_URL %>favicon.ico">
8    <title><%= htmlWebpackPlugin.options.title %></title>
9  </head>
10 <body>
11   <noscript>
12     <strong>We're sorry but <%= htmlWebpackPlugin.options.title %> doesn't
13   </noscript>
14   <div id="app"></div>
15   <!-- built files will be auto injected -->
16 </body>
17 </html>
```

index.html

```
JS main.js X
my-app > src > JS main.js
1  import { createApp } from 'vue'
2  import App from './App.vue'
3
4  createApp(App).mount('#app')
5
```

main.js

แผนภาพการทำงาน

```
<template>
  
  <h1>KongRuksiam Studio</h1>
</template>
```

```
<script>

export default {
  name: 'App',
}
</script>
```



App.vue

```
index.html X
my-app > public > index.html > ...
1  <!DOCTYPE html>
2  <html lang="">
3  <head>
4    <meta charset="utf-8">
5    <meta http-equiv="X-UA-Compatible" content="IE=edge">
6    <meta name="viewport" content="width=device-width,
7    <link rel="icon" href="%= BASE_URL %>favicon.ico"
8    <title>%= htmlWebpackPlugin.options.title %</tit
9  </head>
10 <body>
11   <noscript>
12     <strong>We're sorry but <%= htmlWebpackPlugin.opt
13   </noscript>
14   <div id="app"></div>
15   <!-- built files will be auto injected -->
16 </body>
17 </html>
```



index.html

```
JS main.js X
my-app > src > JS main.js
1  import { createApp } from 'vue'
2  import App from './App.vue'
3
4  createApp(App).mount('#app')
5
```

main.js

JS

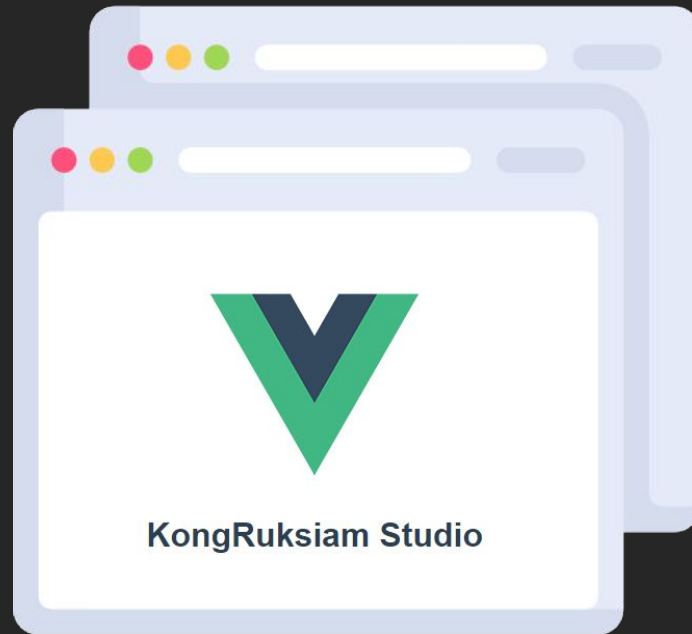
แผนภาพการทำงาน

```
<template>
  
  <h1>KongRuksiam Studio</h1>
</template>

<script>

export default {
  name: 'App',
}
</script>
```

```
JS main.js x
my-app > src > JS main.js
1 import { createApp } from 'vue'
2 import App from './App.vue'
3
4 createApp(App).mount('#app')
5
```



Virtual DOM



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



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

รู้จักกับ Vue.js



Vue.js คือ Frontend Framework

ที่ใช้สำหรับสร้าง User Interface (UI) หรือ
หน้าจอของแอปพลิเคชันในรูปแบบไดนามิก
ที่สามารถอัปเดตบางส่วนของแอปพลิเคชัน
โดยไม่ต้องโหลดแอปพลิเคชันขึ้นมาใหม่
ทั้งหมด



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



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

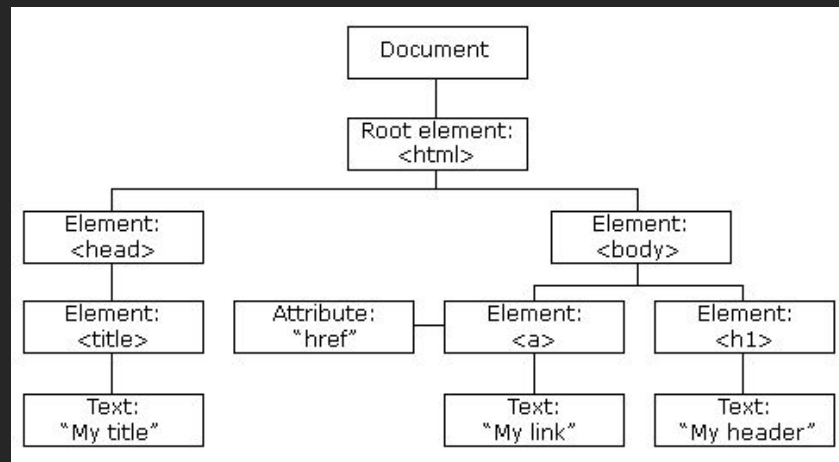
จุดเด่นของ Vue.js

- มีขนาดเล็ก **ทำงานได้อย่างรวดเร็ว**
- มีโครงสร้างการเขียนที่ไม่ซับซ้อน
- แบ่งแอปพลิเคชันออกเป็นส่วนย่อยๆ หรือเรียกว่า Component ง่ายต่อการปรับปรุงแก้ไขและพัฒนาระบบ
- เป็นรูปแบบ Dynamic Webpage เนื้อหาในหน้าเว็บมีการเปลี่ยนแปลงตามเงื่อนไขที่กำหนด

DOM HTML (RealDOM)

เมื่อเข้าสู่หน้าเว็บไซต์และโหลดเสร็จเรียบร้อยแล้ว Web Browser มันจะสร้าง DOM ของหน้านั้นขึ้นมา โดยมอง HTML เป็นโครงสร้างต้นไม้ (ก้อน Object) หรือ**เรียกว่า DOM (Document Object Model)**

```
<html>
<head>
  <title>My title</title>
</head>
<body>
  <a href="#">My link</a>
  <h1>My header</h1>
</body>
</html>
```



Tag ต่าง ๆ ใน HTML จะเรียกว่า Element

Virtual DOM

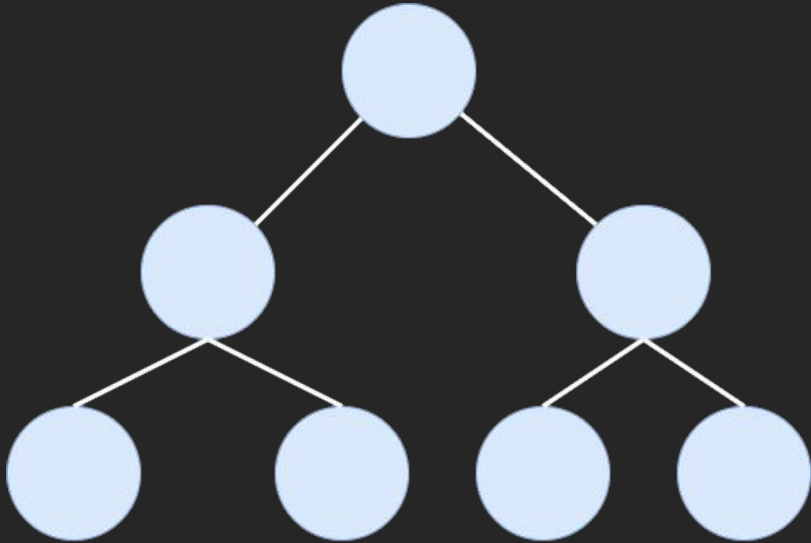
การทำงานของ Virtual DOM

Virtual DOM มีลักษณะคล้ายๆ กับ DOM ใน HTML โดย Virtual DOM จะทำการคัดลอก DOM จริง (Real DOM) มาเก็บลง Memory ถ้ามีการเปลี่ยนแปลงเกิดขึ้นที่ Component ก็จะอัปเดตเฉพาะ Component ที่เปลี่ยนแปลงเท่านั้น โดยไม่จำเป็นต้องอัปเดต DOM ใหม่หมดทั้งหน้า ทำให้เกิดการทำงานได้อย่างรวดเร็ว

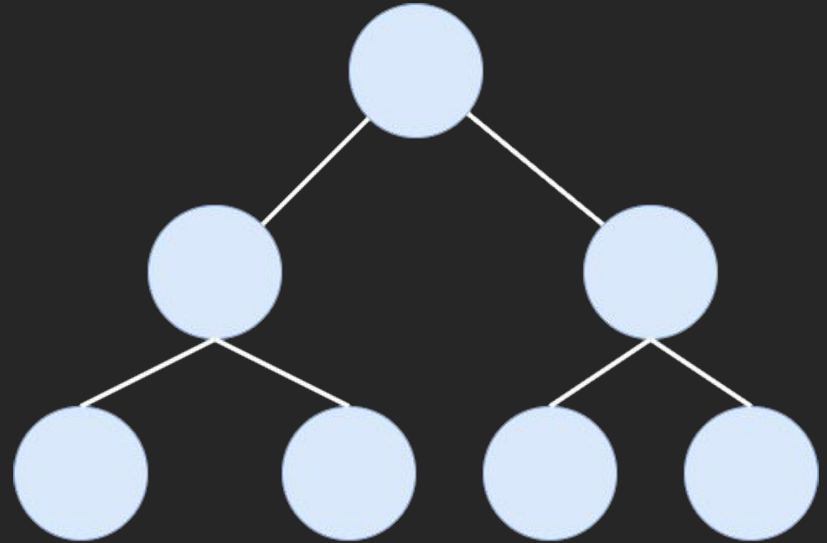
***ถ้าใช้ (DOM แบบปกติจะ Refresh ทั้งหน้าเพื่ออัปเดตหน้าเว็บที่เปลี่ยนแปลงไป)

Virtual DOM

Virtual DOM

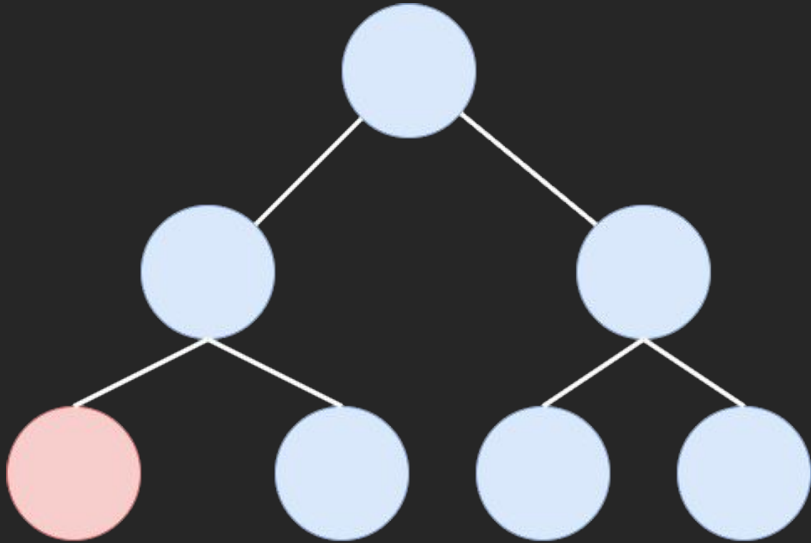


Real DOM

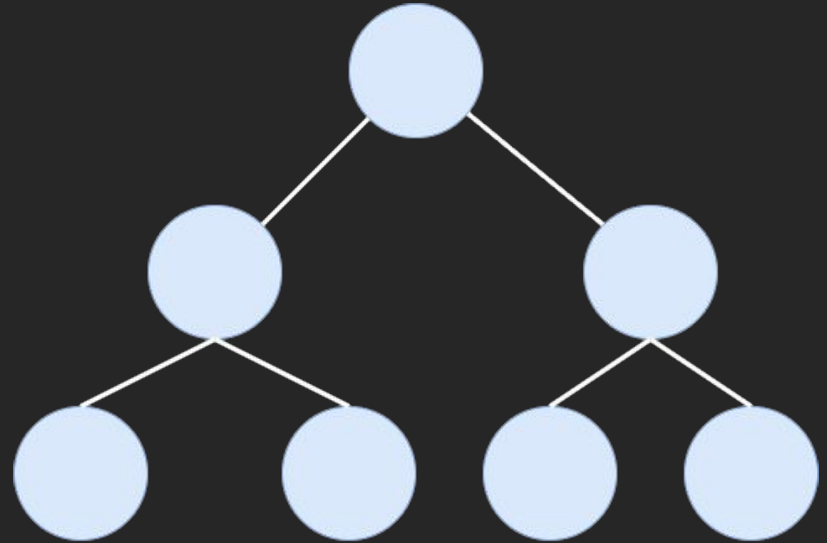


Virtual DOM

Virtual DOM

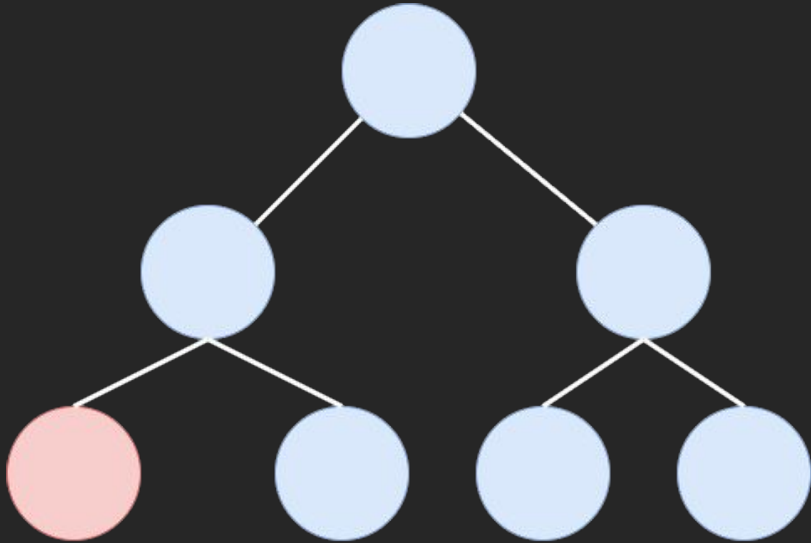


Real DOM

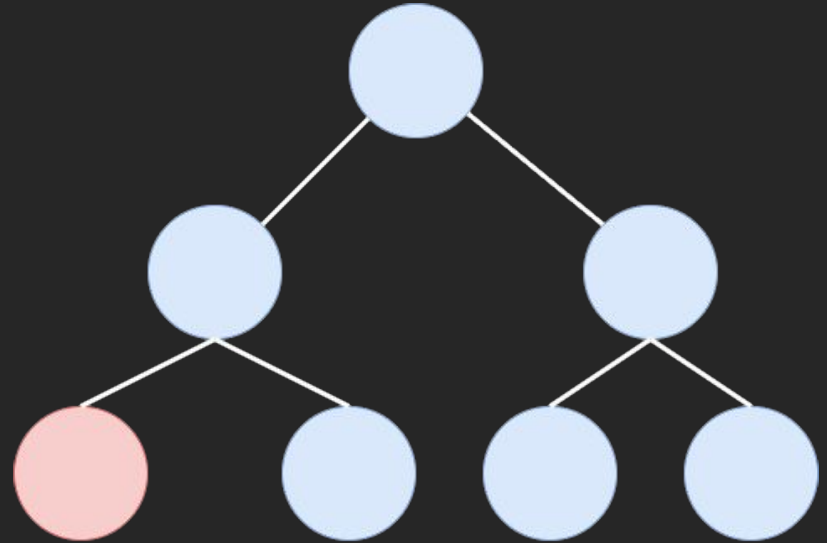


Virtual DOM

Virtual DOM

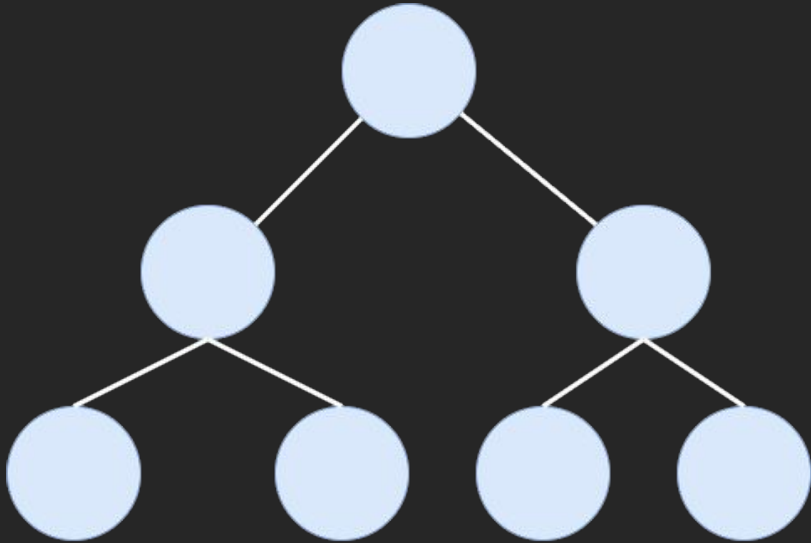


Real DOM

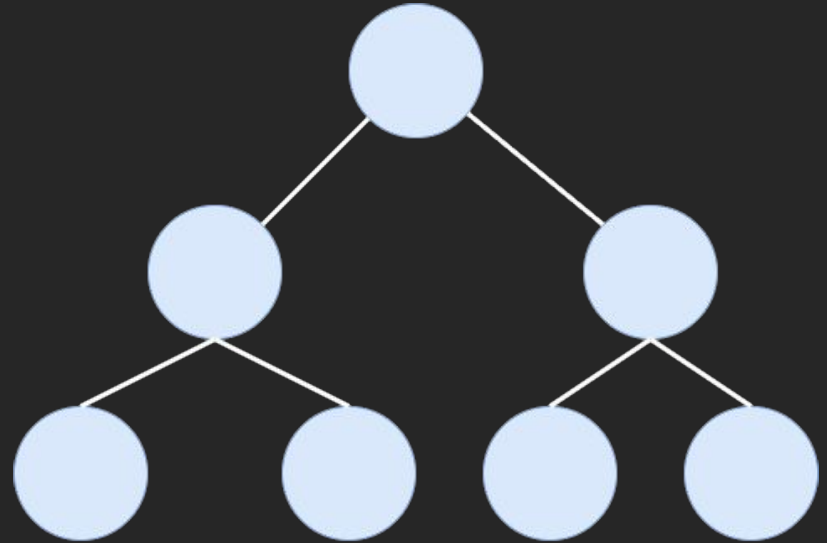


Virtual DOM

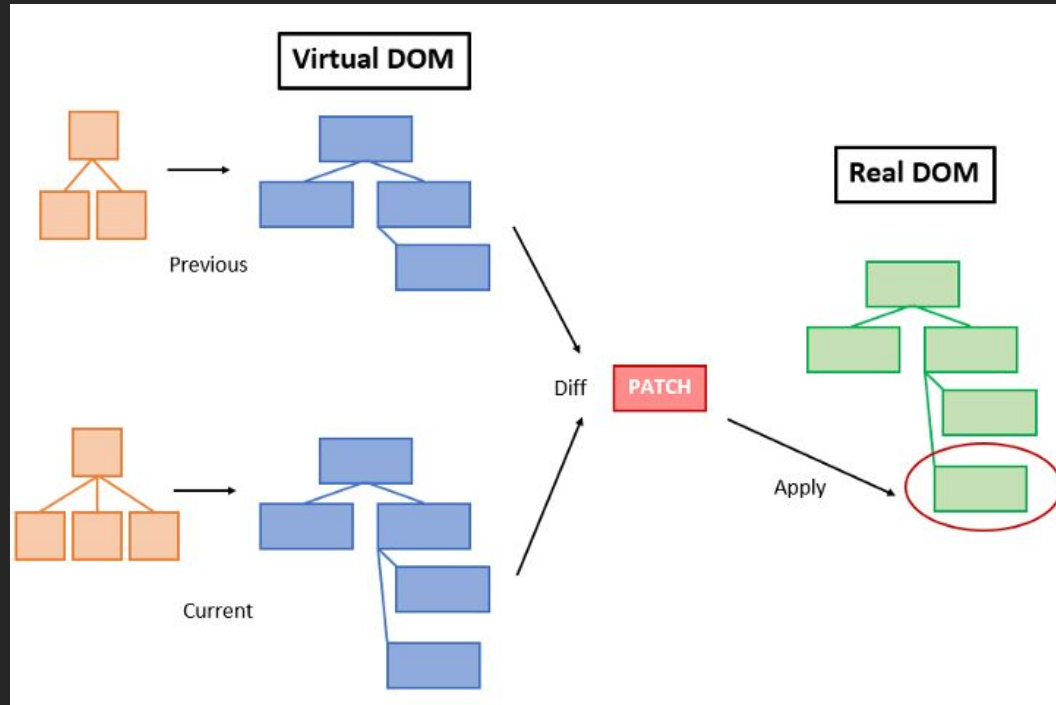
Virtual DOM



Real DOM



Virtual DOM



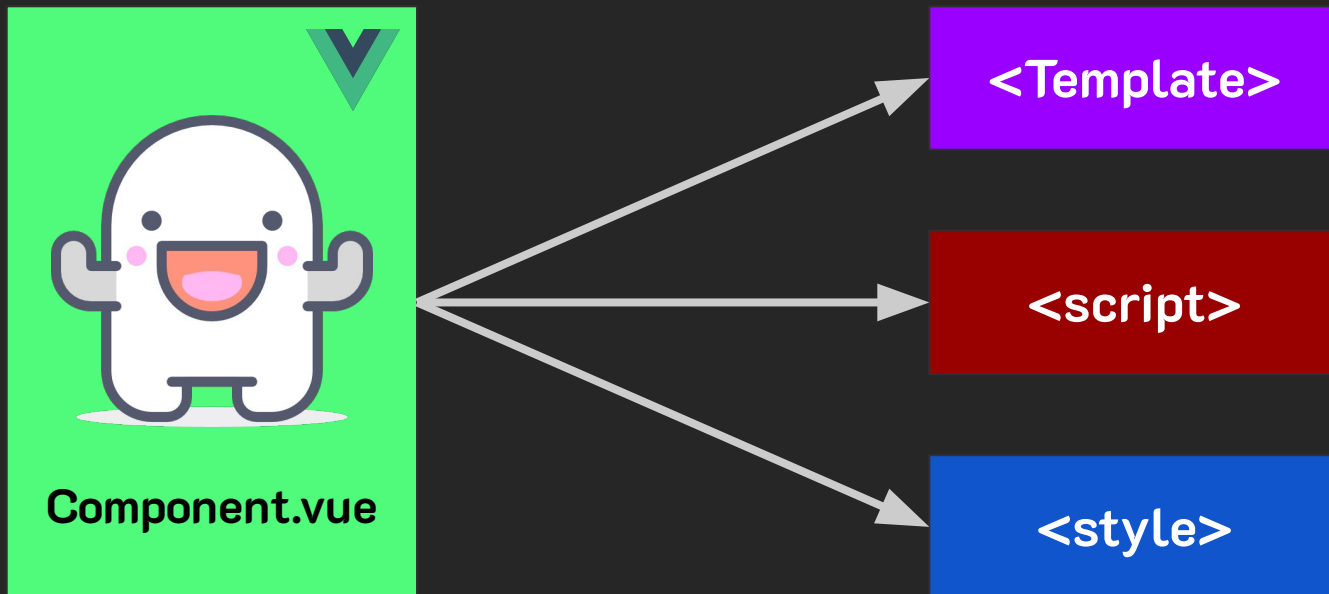
พื้นฐานเกี่ยวกับ Component

พื้นฐานเกี่ยวกับ Component

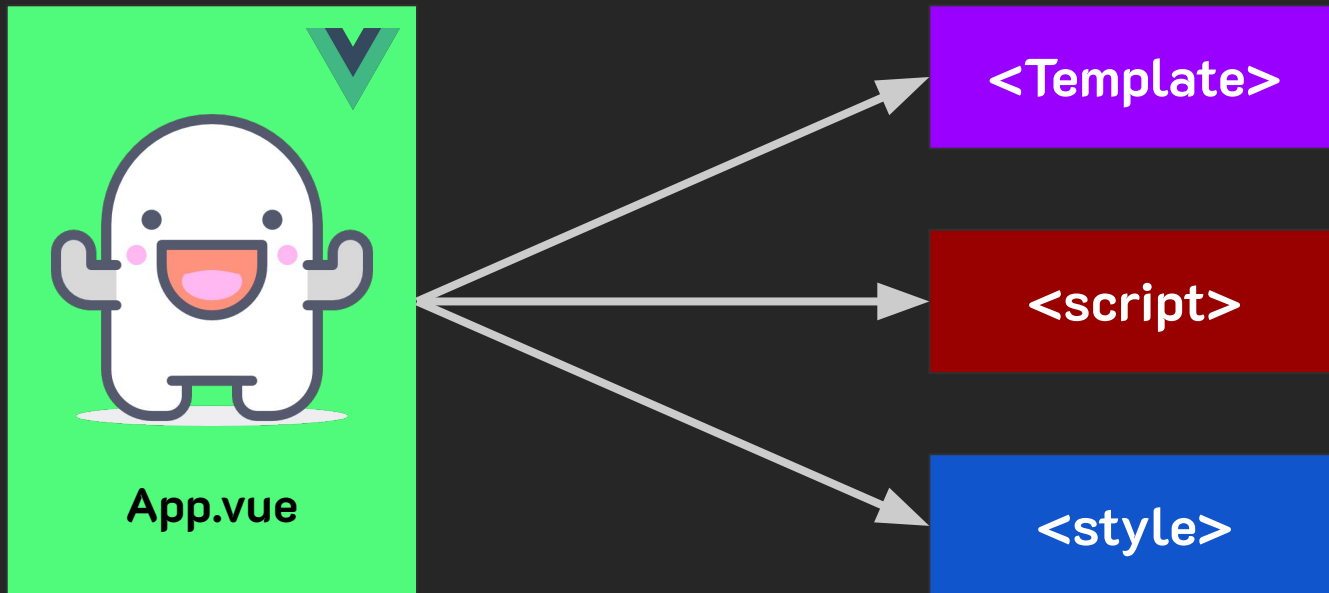
Vue จะแยกแอปพลิเคชันออกเป็นส่วนย่อยๆ ที่เรียกว่า Component โดย Component ใน Vue ก็คือไฟล์ที่มีนามสกุล `.vue` ซึ่งโครงสร้างภายในไฟล์ดังกล่าวจะประกอบด้วย 3 องค์ประกอบหลักๆ ดังนี้

- แท็ก `<template></template>` สำหรับแสดงผล
- แท็ก `<script></script>` สำหรับเขียนคำสั่ง javascript เพื่อควบคุมการทำงานของ Component เช่น การกำหนด data , method เป็นต้น
- แท็ก `<style></style>` เขียน css เพื่อออกแบบและตกแต่ง template

องค์ประกอบของ Component



องค์ประกอบของ Component



องค์ประกอบของ Component

```
<template>
  
  <h1>KongRuksiam Studio</h1>
</template>
```

<Template>

```
<script>
export default {
  name: 'App',
}
</script>
```

<script>

```
<style>
#app {
  font-family: Avenir, Helvetica, Arial, sans-serif;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
  text-align: center;
  color: #2c3e50;
  margin-top: 100px;
}
</style>
```

<style>



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



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

กำหนดข้อมูลใน Component ด้วยฟังก์ชัน data



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



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

โครงสร้างคำสั่ง

```
<script>  
export default{  
  data(){  
    return{ properties : value }  
  }  
}  
</script>
```

data() จะส่งค่ากลับมาในรูปแบบ Object โดยค่าหรือข้อมูลที่ส่งกลับมานั้นสามารถนำไปใช้ใน template หรือ ส่วนอื่นๆของ Component ได้



Interpolation



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



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

Method



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



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

รูปแบบการอ้างอิง Properties

- อ้างอิงใน `<template>` จะใช้ interpolation `{}`
- อ้างอิงใน `<script>` จะได้ keyword : `this`

รูปแบบการอ้างอิง Method

- อ้างอิงใน `<template>` จะใช้ interpolation { ชื่อ method }
- อ้างอิงใน `<script>` จะได้ keyword : `this.ชื่อ method`



Event Modifier

เป็นการกำหนดค่าหรือแก้ไข Event ที่สนใจ เช่น @click อยาก
เช็คว่าการคลิกเมาส์ซ้ายหรือคลิกเมาส์ขวา หรือมีการใช้งานร่วม
กับการกด key ต่างๆ ไปจนถึงการทำงานในแบบฟอร์มสำหรับดัก
Event ตอนกดปุ่ม Submit เป็นต้น



Ref Access DOM Element



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



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

Render Condition (v-if,v-else)

v-if เป็น directive สำหรับการตรวจสอบเงื่อนไขว่าเป็นจริงหรือเท็จ ซึ่งมีผลกับการเปลี่ยนแปลงโครงสร้างของ DOM โดย

- ถ้าเงื่อนไขเป็นจริง ก็จะนำเอา Element ไปแทรกลงใน DOM
- ถ้าเป็นเท็จก็จะไม่แทรก Element ลงไป



List Data ด้วย v-for (Loop)

v-for เป็น directive สำหรับเพิ่ม Element ไปยัง DOM ซึ่งจะเข้าไปที่สมาชิกทุกๆตัวของ Array หรือ Object

การใช้งาน v-for ต้องใส่ key ที่ไม่ซ้ำลงไปด้วย สำหรับการอ้างอิง Element และเพื่อให้ง่ายต่อการเพิ่ม-ลบ Element ในภายหลัง



ซ่อน/แสดง Element (v-show)

v-show เป็น directive สำหรับการตรวจสอบเงื่อนไขว่าเป็นจริงหรือเท็จ สำหรับซ่อนและแสดง Element โดย

- ถ้าเงื่อนไขเป็นจริง จะแสดง Element
- ถ้าเงื่อนไขเป็นเท็จ จะซ่อน Element

ซ่อน/แสดง Element (v-show)

v-show จะแตกต่างจาก v-if ตรงที่ v-show จะถูก Render และแสดงไปยัง DOM เรียบร้อยแล้ว ซึ่งแตกต่างจาก v-if ที่จะต้องแทรก Element ลงไปใน DOM ในภายหลัง แต่จะใช้วิธีการตรวจสอบผ่านเงื่อนไขที่กำหนดขึ้นมาเหมือนกันเท่านั้น!

Computed Properties



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



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

Computed Properties

นอกจาก Properties แบบทั่วไปที่ใช้เก็บข้อมูลภายใน Component แล้ว ยังมี Properties อีกรูปแบบหนึ่งที่ทำงานใน Vue.js ที่มีชื่อว่า Computed Properties สำหรับเก็บค่าที่ได้จากการคำนวณ หรือ เก็บค่าที่ได้จากการเปลี่ยนแปลงค่าใน Properties อื่นๆ

Computed Properties

Computed Properties จะนำมาใช้ในการจัดการเกี่ยวกับข้อมูลที่เราสนใจภายใน Component โดย Computed Properties จะทำงานเมื่อข้อมูลที่เราสนใจมีการเปลี่ยนแปลงเกิดขึ้น



เปรียบเทียบ Method VS Computed



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



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

Computed Properties

การนิยาม Computed Properties จะมีลักษณะคล้ายๆกับ Method

แต่โครงสร้างการทำงานจะต้องมีการ Return ค่าเสมอ (Getter Method)

แต่สิ่งที่แตกต่างระหว่าง Method กับ Computed คือ

- Computed จะทำการ Caching ค่าเอาไว้ เพื่อตัดการทำงานเมื่อข้อมูลที่เราสนใจมีการเปลี่ยนแปลงค่า
- Method จะถูกเรียกใช้ทุกครั้งที่ Component Re-Render ใหม่ ส่วน Computed จะถูกเรียกใช้เมื่อข้อมูลเปลี่ยนแปลงค่า



การนำ Computed Properties ไปใช้งาน

- กรองข้อมูล
- คำนวณ
- กำหนดเงื่อนไข
- ดึงข้อมูล (Getter Method)



Watchers



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



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

Watchers

นอกจากการใช้งาน Computed Properties ในการเก็บผลลัพธ์จากการเปลี่ยนแปลงข้อมูลที่เราสนใจแล้ว

ใน Vue.js ยังมี ส่วนที่เรียกว่า **Watchers** สำหรับติดตามการเปลี่ยนแปลงข้อมูลที่เกิดขึ้นใน Component

Watchers

เมื่อข้อมูลที่เราสนใจมีการเปลี่ยนแปลงค่า Watcher จะ
รับหน้าที่ติดตามการเปลี่ยนแปลงที่เกิดขึ้น โดยจะเรียกใช้
งาน Method ที่กำหนดขึ้นมาเมื่อข้อมูลที่ติดตามนั้นมีการ
เปลี่ยนแปลงค่าไป



Watchers

Watchers นั้นสามารถทำงานกับคำสั่งที่อยู่ในรูปแบบ Asynchronous ได้ นั่นหมายความว่าการทำงาน Watchers มาใช้งานจะสามารถติดตามการเปลี่ยนแปลง เมื่อมีการหน่วงเวลาเกิดขึ้นกับข้อมูลที่น่าสนใจ เช่น ดึงข้อมูลจาก Server โดยใช้ระยะเวลาดึงข้อมูลประมาณ 2 วินาที ถ้าดึงข้อมูลจนครบก็นำข้อมูลมาใช้งาน เราสามารถนำเอา Watchers มาติดตามได้ว่า ดึงข้อมูลมาครบหรือไม่ ถ้าดึงครบแล้วจะให้ทำอะไรต่อไป ?



Watchers VS Computed Properties

1. Watchers กับ Computed Properties นั้นจะมีคล้ายคลึงกัน คือ ทำงานกับข้อมูลที่เราสนใจ
2. Watchers ติดตามการเปลี่ยนแปลงที่เกิดขึ้นกับข้อมูล ส่วน Computed Properties เก็บผลลัพธ์การเปลี่ยนแปลงที่เกิดขึ้น
3. Watchers ทำงานกับกลุ่มคำสั่งแบบ Asynchronous ได้ ส่วน Computed Properties ทำงานกับกลุ่มคำสั่งแบบ Asynchronous ไม่ได้



END PHASE 1



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



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

1. พื้นฐานเกี่ยวกับ Component

Component

คอมโพเนนต์ (Component) คือ ส่วนประกอบย่อยของแอปพลิเคชัน สำหรับใช้แสดงผลลัพธ์และรับคำสั่งต่างๆ จากผู้ใช้ นอกจากนี้ Component ยังสามารถแชร์คุณสมบัติหรือความสามารถบางอย่างไปให้ Component อื่นได้ด้วย

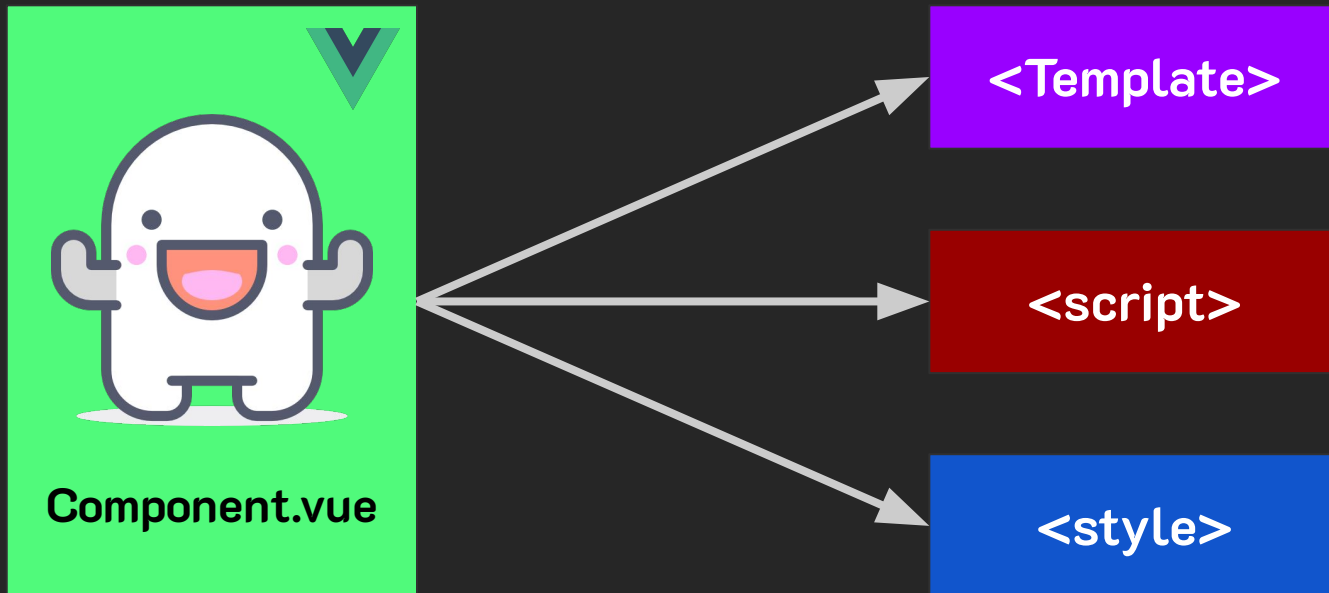


พื้นฐานเกี่ยวกับ Component

Vue จะแยกแอปพลิเคชันออกเป็นส่วนย่อยๆ ที่เรียกว่า Component โดย Component ใน Vue ก็คือไฟล์ที่มีนามสกุล `.vue` ซึ่งโครงสร้างภายในไฟล์ดังกล่าวจะประกอบด้วย 3 องค์ประกอบหลักๆ ดังนี้

- แท็ก `<template></template>` สำหรับแสดงผล
- แท็ก `<script></script>` สำหรับเขียนคำสั่ง javascript เพื่อควบคุมการทำงานของ Component เช่น การกำหนด data , method เป็นต้น
- แท็ก `<style></style>` เขียน css เพื่อออกแบบและตกแต่ง template

องค์ประกอบของ Component



องค์ประกอบของ Component

```
<template>
  
  <h1>KongRuksiam Studio</h1>
</template>
```

<Template>

```
<script>
export default {
  name: 'App',
}
</script>
```

<script>

```
<style>
#app {
  font-family: Avenir, Helvetica, Arial, sans-serif;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
  text-align: center;
  color: #2c3e50;
  margin-top: 100px;
}
</style>
```

<style>

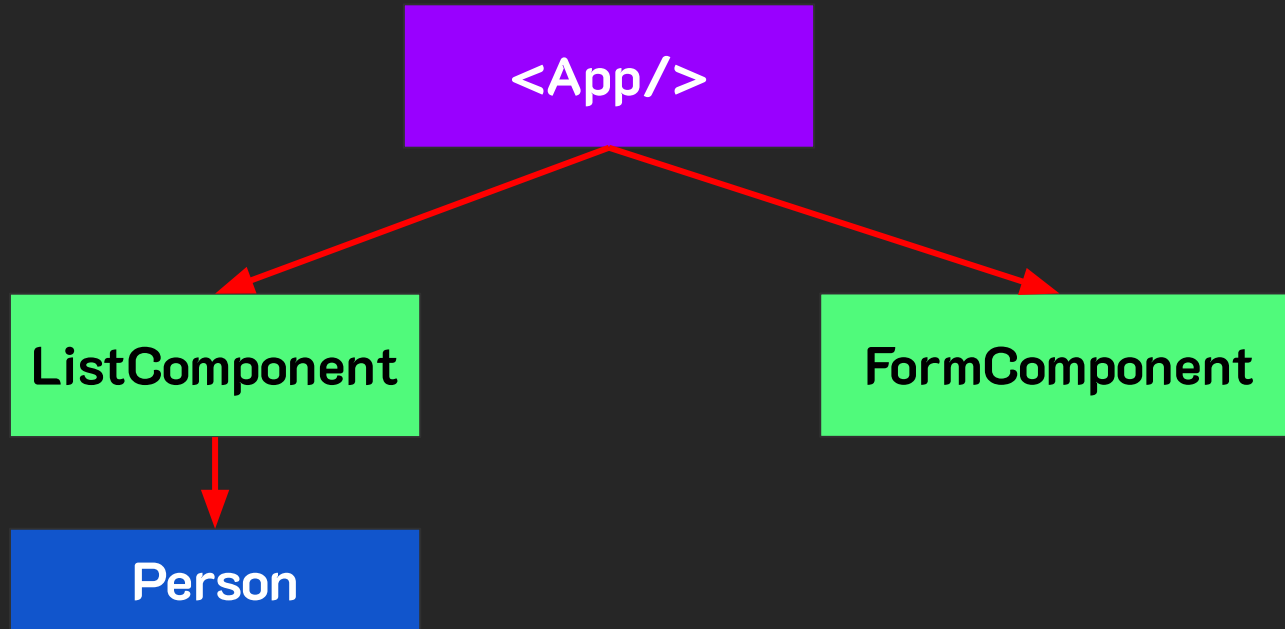


<https://www.youtube.com/c/KongRuksiamOfficial/>

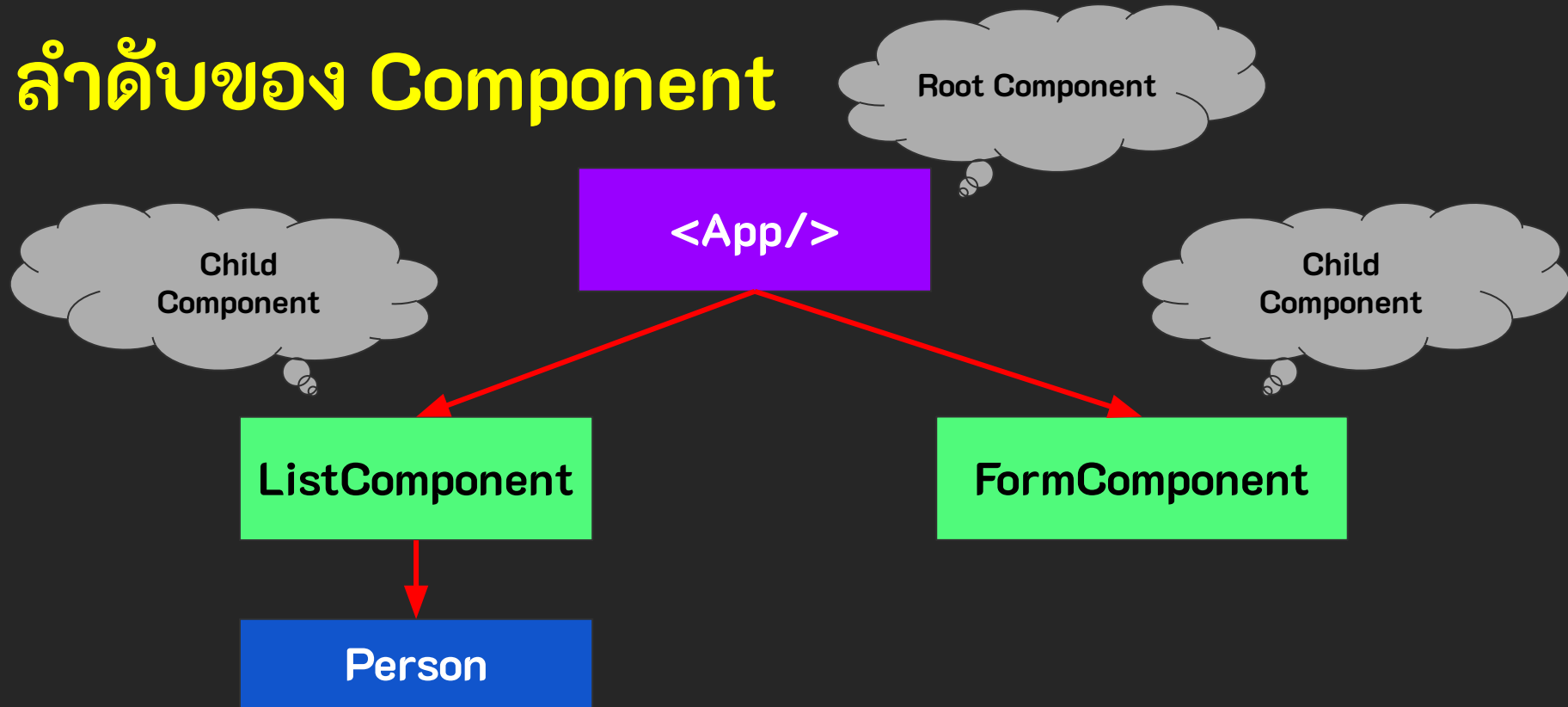


<https://www.facebook.com/KongRuksiamTutorial/>

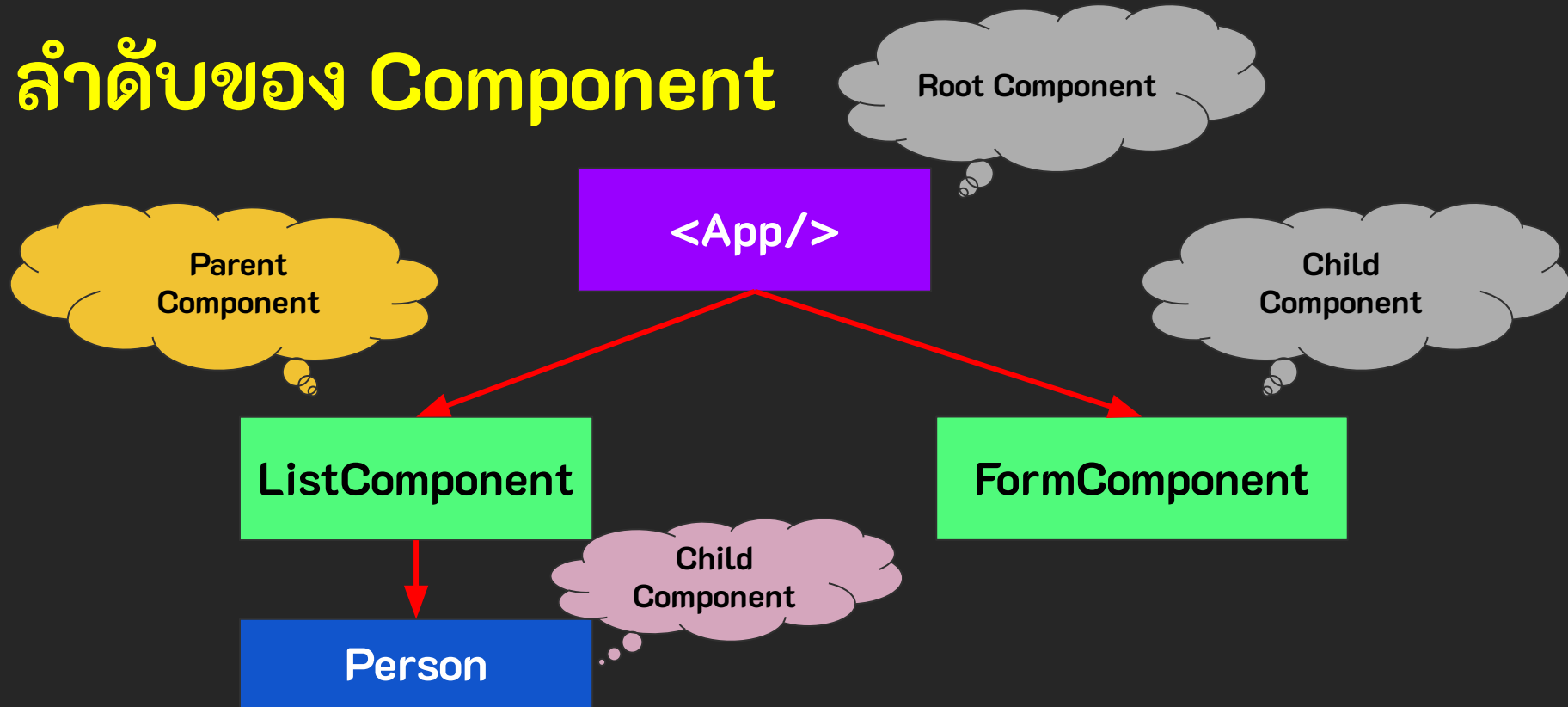
ลำดับของ Component



ลำดับของ Component



ลำดับของ Component



2.สร้าง Component



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



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

สร้าง Component



- สร้าง Component และจัดเก็บในโฟลเดอร์ Components
- การ Export Component ไปทำงานใน Component อื่นๆ
- การเรียกใช้งาน Component



3.Style Scoped



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



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

Style Scoped

โดยปกติ เมื่อกำหนด CSS ลงใน `<style></style>` ที่ทำงานใน Component จะส่งผลให้ทุกๆ Component นั้นใช้ style แบบเดียวกัน ถ้าต้องการให้ค่า CSS มีผลเฉพาะ Component ปัจจุบันที่เราสนใจจะใช้ `scoped` แทรกลงไปใน `<style>`



โครงสร้างคำสั่ง

```
<style scoped>
```

```
....
```

```
....
```

```
</style>
```



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



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

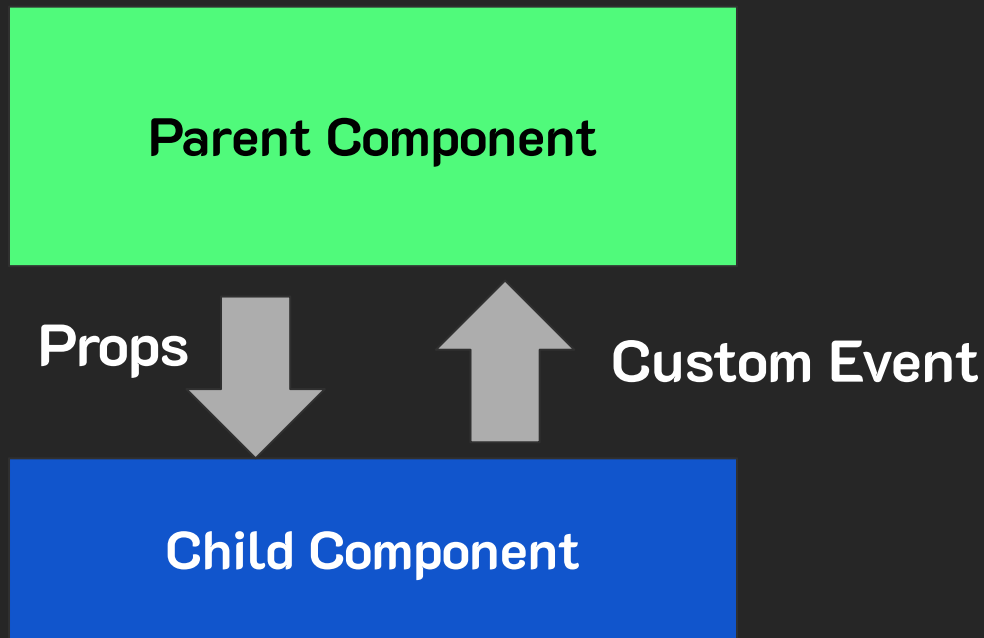
4.การส่งข้อมูลระหว่าง Components

การส่งข้อมูลระหว่าง Component

การส่งข้อมูลจาก Component หนึ่งไปสู่อีก Component หนึ่ง
มีอยู่ด้วยกัน 2 รูปแบบ

- Parent to Children ผ่าน Props (Up -> Down)
- Children to Parent ผ่าน Custom Event (Down->Up)

Props และ Custom Event



รู้จักกับ Props

Props คือ ข้อมูลที่สามารถส่งเข้าไปใน Components ได้ โดยข้อมูลดังกล่าวจะถูกส่งจาก Component ที่อยู่สูงกว่า (Parent Component) ไปยัง Component ที่อยู่ต่ำกว่า (Child Component)



รู้จักกับ Props

การส่งค่า Props จาก Parent Component ไปยัง Child Component
มีวัตถุประสงค์ดังนี้

- กำหนดข้อมูลสำหรับแสดงเนื้อหาใน Child Component
- กำหนดความสามารถบางอย่างให้กับ Child Component

ในการติดต่อหรือสื่อสารข้อมูลหากันระหว่าง Component



รู้จักกับ Props

Props แบบค่าเดียว

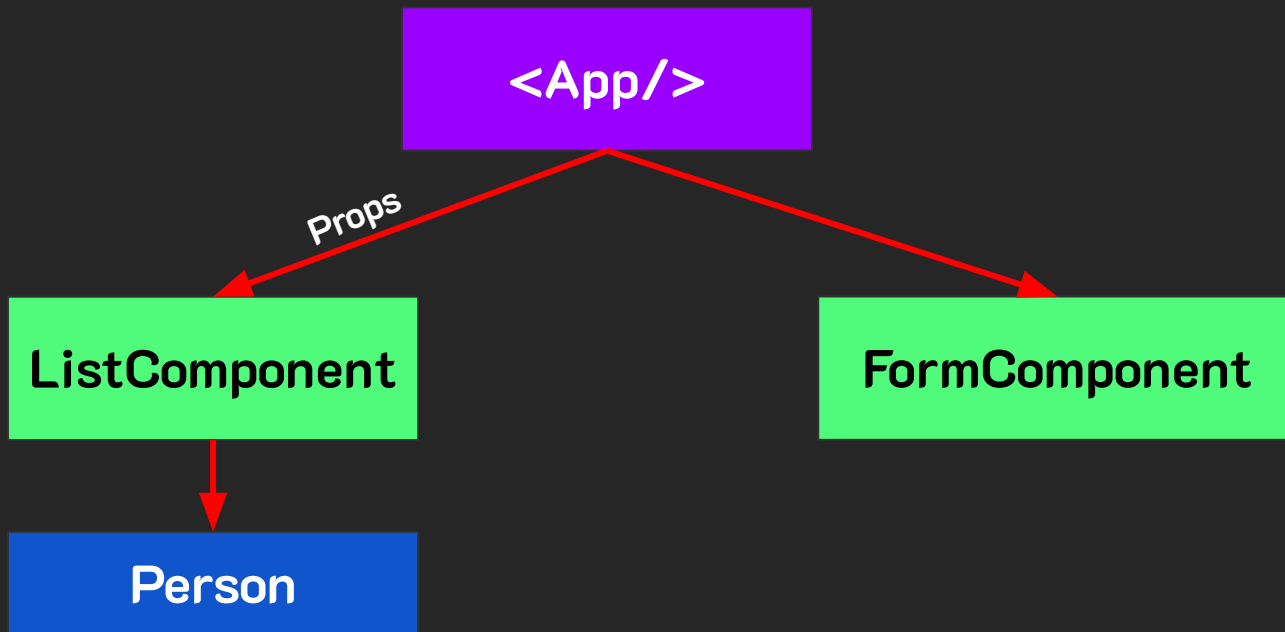
<ชื่อคอมโพเนน ชื่อพร็อพ =ค่าที่กำหนดให้พร็อพ/>

Props แบบหลายค่า

<ชื่อคอมโพเนน ชื่อพร็อพ =ค่าที่1 ชื่อพร็อพ =ค่าที่2 />



รู้จักกับ Props



การรับค่า Props ใน Child Component



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



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

โครงสร้างคำสั่ง

```
export default {  
  name : "ชื่อ component",  
  props: ["ชื่อ prop"]  
};
```



5.Dynamic Props



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



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

6.Prop Validation



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



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

โครงสร้างคำสั่ง

```
export default {
```

```
  props:{
```

```
    propName : role / optional
```

```
  }
```

```
};
```

example

- type
- required
- default



7.Style Component



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



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

8.Custom Event

(Children to Parent)

Custom Event

ในกรณีที่ Child Component ต้องการส่งข้อมูลหรือติดต่อกลับไปยัง Parent Component จะไม่ใช้วิธีการแบบ Prop แต่จะใช้วิธีการปล่อย Event กลับไปบอกกับ Parent Component แทน

เมื่อ Parent Component ตรวจสอบว่ามี Event ส่งมาจาก Child Component ก็จะรับเอาข้อมูลที่ส่งมาพร้อมกับ Event นั้น มาใช้งาน



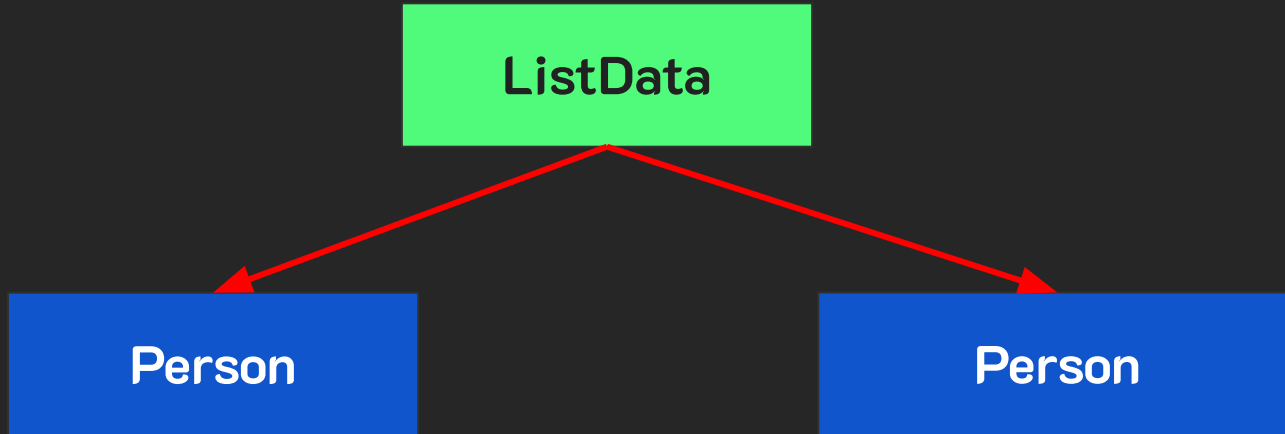
Custom Event

Parent Component ต้องกำหนด Event Listener สำหรับรอรับ Event จาก Child Component จะเรียนส่วนนี้ว่า “Custom Event”

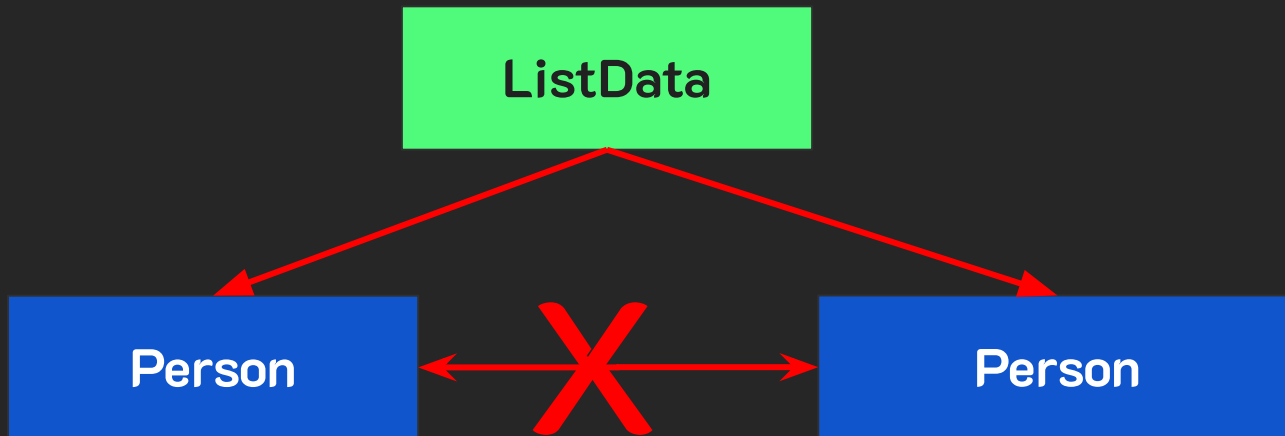
เมื่อ Child Component ต้องการติดต่อไปหา Parent Component ก็จะทำการปล่อย (emit) Event พร้อมส่งข้อมูลมายัง Parent Component



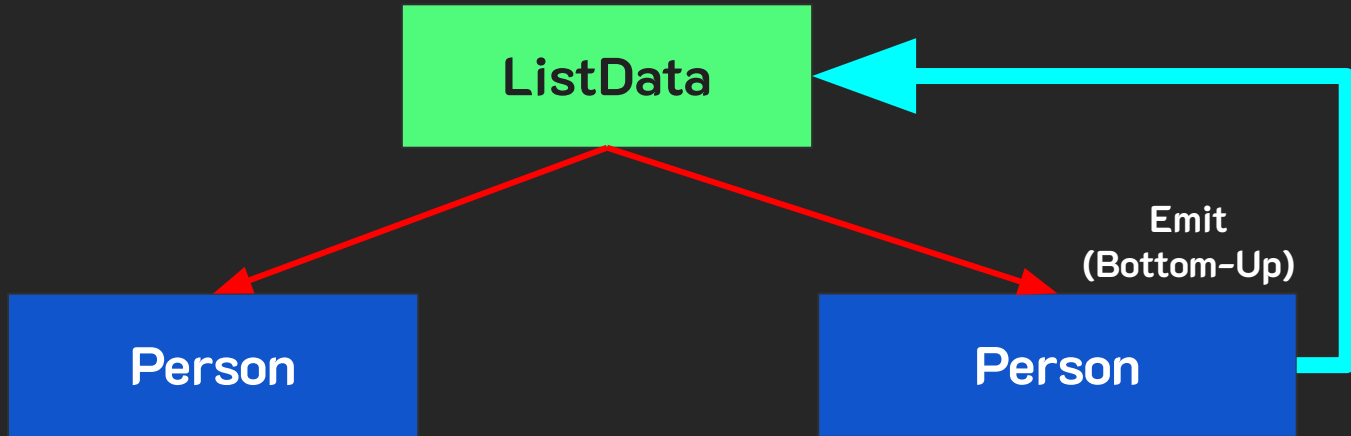
Custom Event (Bottom-Up)



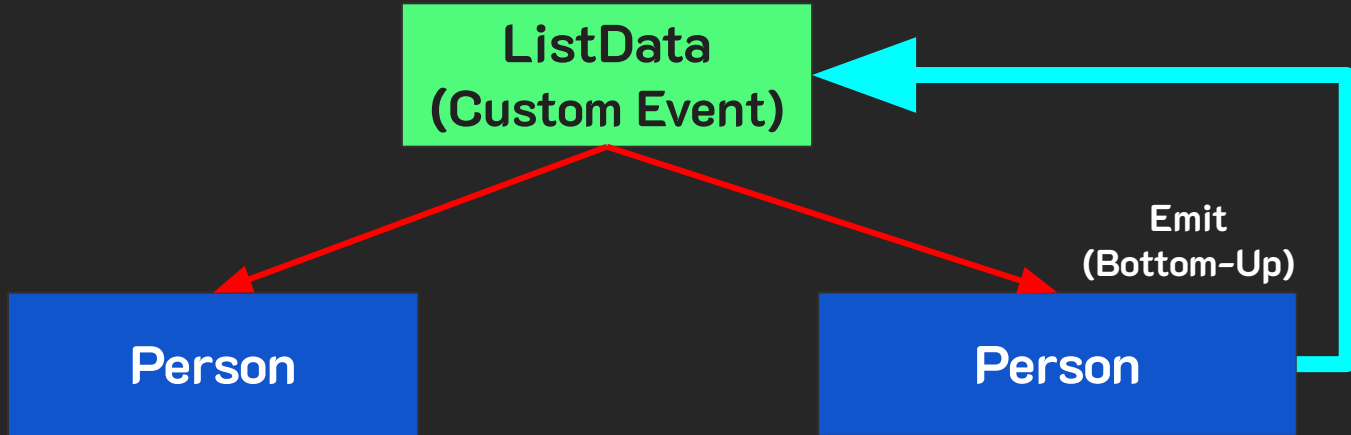
Custom Event (Bottom-Up)



Custom Event (Bottom-Up)



Custom Event (Bottom-Up)



9.Custom Event

(ลบข้อมูล)



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



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

10.Transition



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



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

11.Slot



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



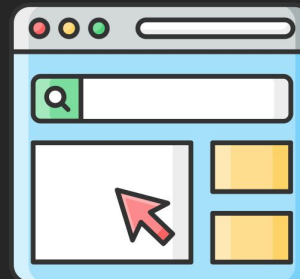
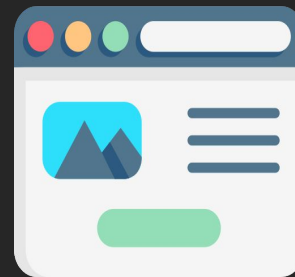
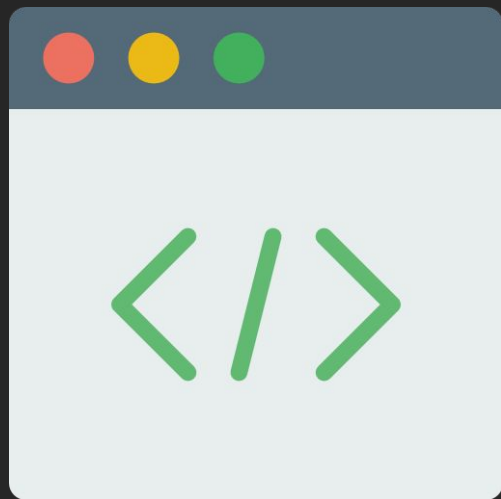
<https://www.facebook.com/KongRuksiamTutorial/>

Slot คืออะไร

Slot ถูกนำมาทำงานร่วมกับการแสดงผลใน Vue Application โดย
จะการสร้างต้นแบบหรือพื้นที่ของ Template ที่บรรจุเนื้อหาในด้านในแตกต่างกัน ส่งผลให้ Component มีความแตกต่างหลากหลายมากขึ้น

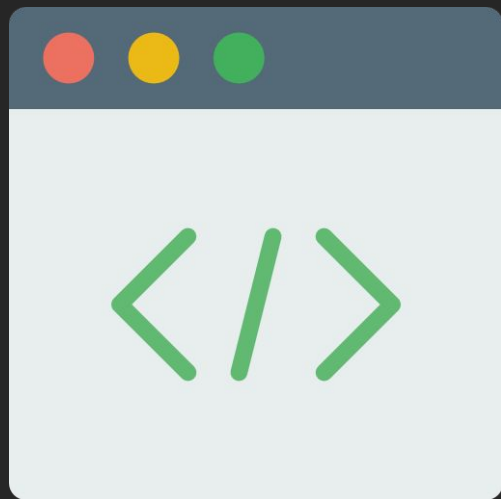


ตัวอย่าง



Component ต้นแบบ

ตัวอย่าง



Component ต้นแบบ



Slot



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



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

12.Name Slots



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



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

13. รู้จักกับ Two-Way Data Binding

Data Binding คืออะไร

การเชื่อมโยงข้อมูลระหว่างส่วนควบคุม (Script) กับส่วนแสดงผล (Template) ให้ทำงานร่วมกันได้ ซึ่ง Data Binding สามารถแบ่งออกเป็น 2 รูปแบบ ได้แก่

- เชื่อมโยงแบบทิศทางเดียว (One Way Binding)
- เชื่อมโยงแบบสองทิศทาง (Two Way Binding)



เชื่อมโยงแบบทิศทางเดียว (One Way Binding)



เป็นการสื่อสารแบบทิศทางเดียว
เช่น การนำข้อมูลไปแสดงผล

- การใช้งาน Interpolation
- Attribute Binding
- Event Binding



เชื่อมโยงแบบสองทิศทาง (Two way Binding)



เป็นการสื่อสารแบบสองทิศทาง เช่น
การป้อนข้อมูลและให้แสดงผลข้อมูล
ในช่วงเวลาเดียวกัน เป็นต้น

Two-Way Data Binding (v-model)

ส่วนใหญ่จะการทำงานร่วมกับแบบฟอร์มใน Vue.js โดยเป็นการเชื่อมโยงกระบวนการทำงานแบบ 2 ทิศทาง หรือ Two Way Binding ได้แก่ ส่วนควบคุม (Script) และส่วนแสดงผล (Template) เนื่องจากข้อมูลที่ป้อนในแบบฟอร์มนั้น จะถูกส่งมาทำงานที่ส่วนควบคุมและส่วนควบคุมเองก็สามารถแสดงข้อมูลดังกล่าวกลับไปที่ส่วนแสดงผลได้ในเวลาเดียวกัน



Two-Way Data Binding (v-model)

การเชื่อมโยงระหว่างส่วนควบคุม (Script) กับ ส่วนแสดงผล (Template) จะใช้ Directive ที่มีชื่อว่า v-model

v-model = “propertyName / data”

- **v-model** คือ การกำหนดว่า input ที่ป้อนให้ไปผูกกับ properties หรือ data ที่เราสนใจ เมื่อ input เปลี่ยน properties ก็เปลี่ยนเมื่อ properties เปลี่ยน input ก็เปลี่ยนไปด้วย
- **propertyName** คือ ส่วนที่ใช้สำหรับจัดเก็บข้อมูลจาก Input และสามารถนำไปแสดงผลได้ในเวลาเดียวกัน



ยกตัวอย่างการใช้งาน v-model



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



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

แบบฟอร์มใน Vue.js

- 14 Input Field (ชื่อ , เงินเดือน)
- 15 Select Field (ตำแหน่งงาน)
- 16 Radio (เพศ)
- 17 CheckBox (ภาษา)

18 บันทึกข้อมูล

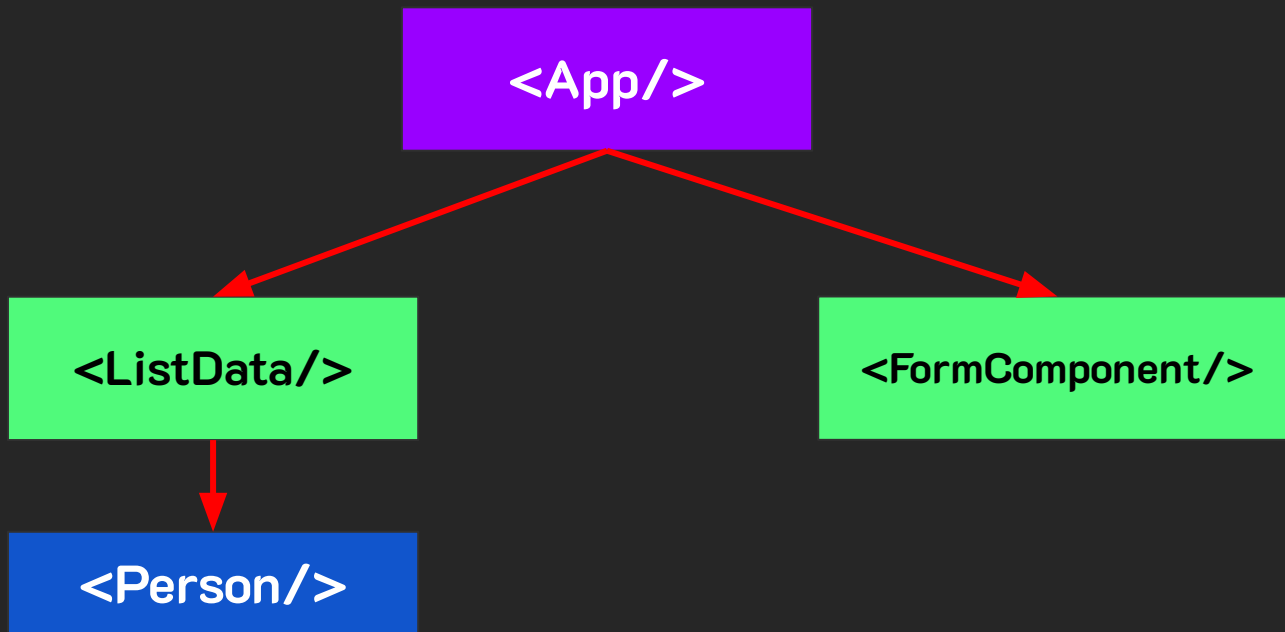


<https://www.youtube.com/c/KongRuksiamOfficial/>

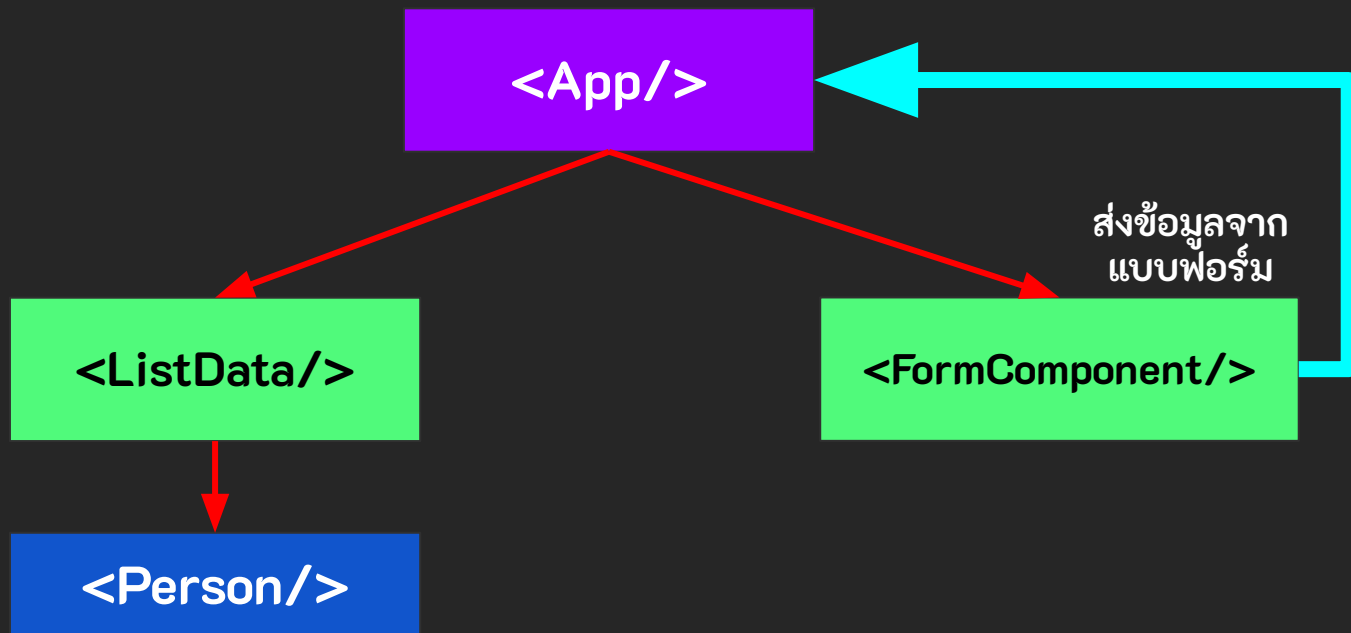


<https://www.facebook.com/KongRuksiamTutorial/>

แผนภาพการทำงาน



แผนภาพการทำงาน



แผนภาพการทำงาน

