

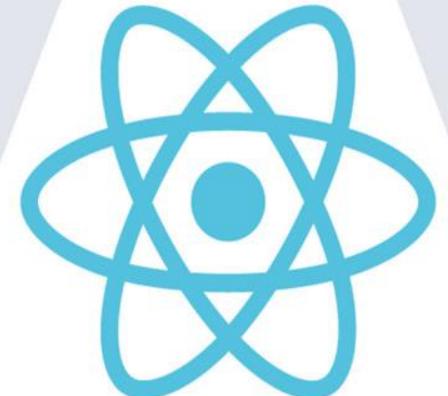
หลักสูตรอบรม

5 DAYS

พัฒนาเว็บแอปพลิเคชันด้วย  
**React v.19**  
และ **NodeJS**  
ทำงานแบบ **docker**



อาจารย์สามิตร  
สถาบันไอทีเนยส์



# วิทยากร



อ.สา米ตร์ โภยม

ปริญญาโทคณะเทคโนโลยีสารสนเทศ  
มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ

## ▶ **Frontend**

Angular, React, NextJS, VueJS,  
Bootstrap, TailwindCSS

## ▶ **Backend**

PHP, Python, Java, NodeJS, ASP.net

## ▶ **Database**

MySQL, PostgreSQL, MS SQL Server,  
MongoDB

## ▶ **Mobile**

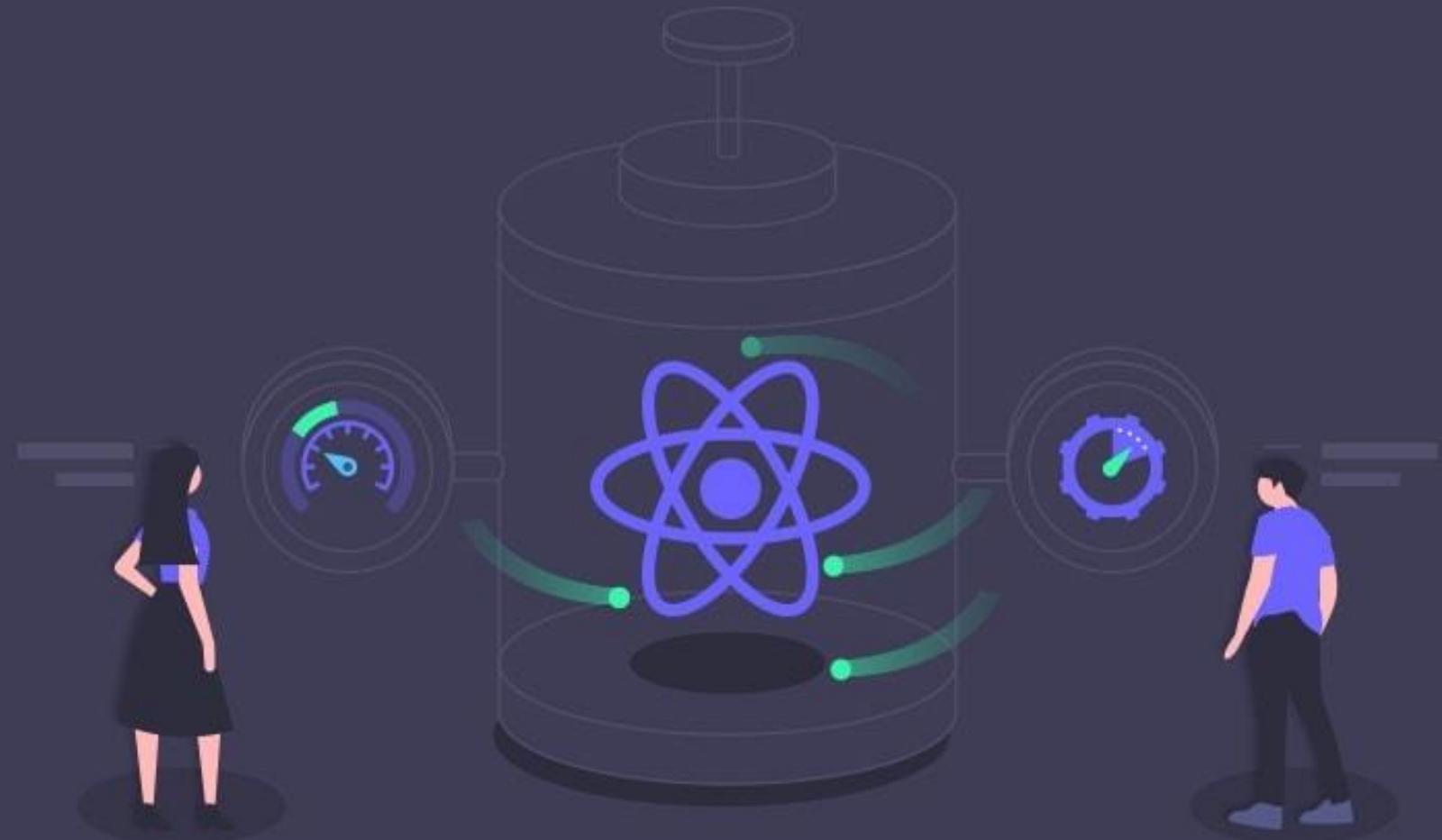
Java, Kotlin, Objective C, Swift, Cordova,  
ionic, Flutter

## ▶ **DevOps**

Git, Github, Gitlab, Docker,  
Kubernetes



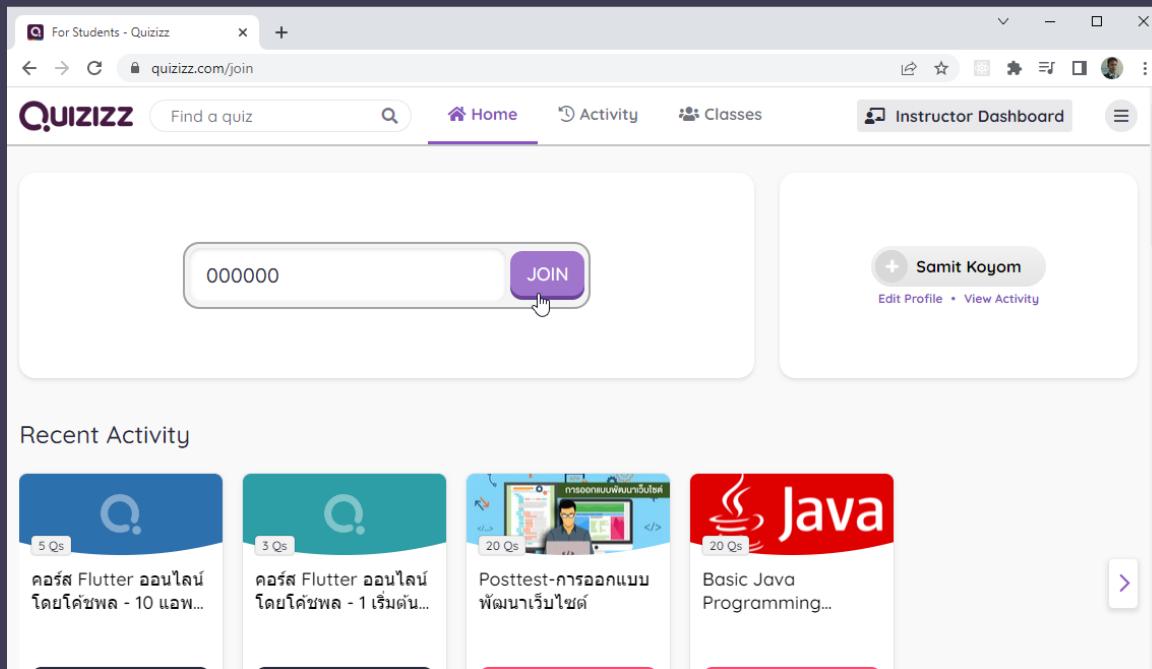
# Pre-Test



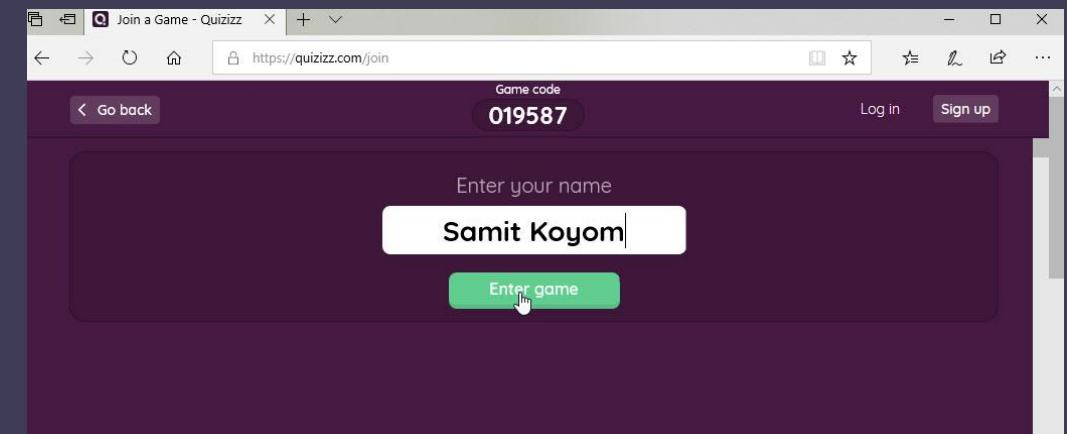
# Pretest ทำแบบทดสอบก่อนอบรม

**STEP 1:** เข้าทำแบบทดสอบที่ลิงก์ ป้อนรหัสเข้าห้องสอบ

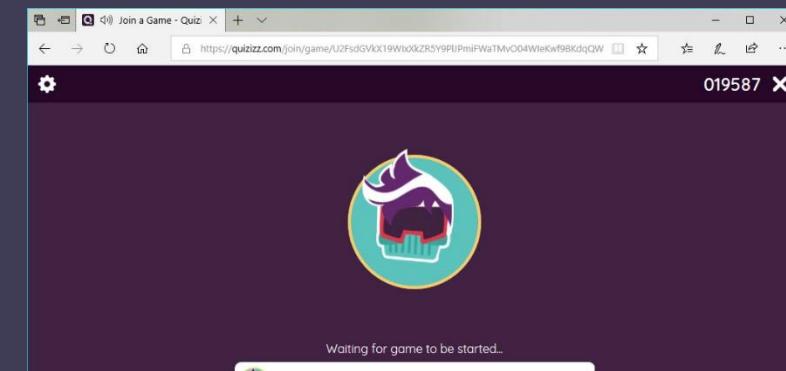
**quizizz.com/join**



**STEP 2:** ป้อนชื่อ



**STEP 3:** รอผู้สอน Start ข้อสอบ



เอกสารประกอบการอบรม  
**bit.ly/rndutac**



# React and NodeJS with Docker Workshop (5 days)

## DAY 1

**Section 1:** ติดตั้งเครื่องมือ เช่น Node.JS, VSCode, Git

**Section 2:** พื้นฐาน HTML 5 & CSS 3 และ JavaScript ES6

**Section 3:** กบกวนพื้นฐาน React.JS

## DAY 2

**Section 4:** สร้าง Workshop โปรเจ็คต์ React 19 ด้วย Vite

**Section 5:** ใช้ React ออกแบบ UI ด้วย Material UI (MUI) และ MUI X ล่าสุด

## DAY 3

**Section 6:** สร้าง Workshop Rest API ด้วย Node.JS และ Express.JS

## DAY 4

**Section 7:** ใช้ React เชื่อมต่อ Web API พร้อมทำ Authentication

## DAY 5

**Section 8:** Workshop React เรียกข้อมูล/เพิ่ม/แก้ไข/ลบ (CRUD) ผ่าน API

**Section 9:** การ Build และ Deployed project ไปใช้งานจริง

# React and NodeJS with Docker Workshop Progress



Day 1



Day 2



Day 3



Day 4



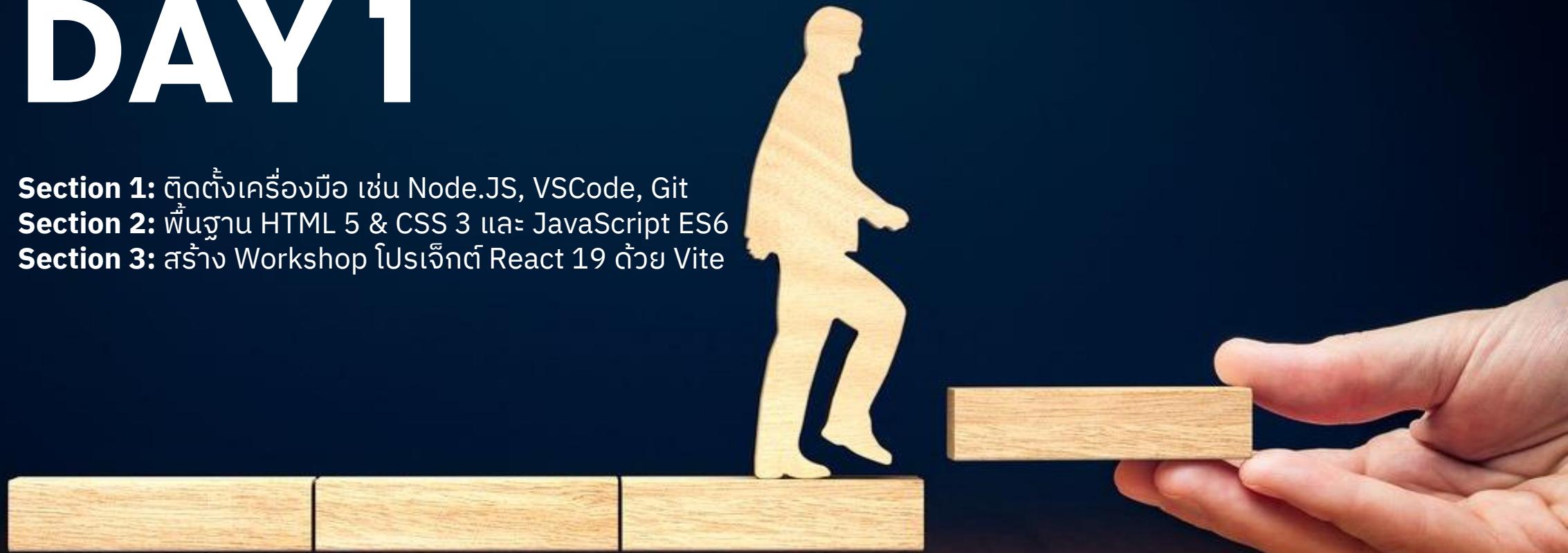
Day 5

# DAY 1

**Section 1:** ติดตั้งเครื่องมือ เช่น Node.js, VSCode, Git

**Section 2:** พื้นฐาน HTML 5 & CSS 3 และ JavaScript ES6

**Section 3:** สร้าง Workshop โปรเจกต์ React 19 ด้วย Vite

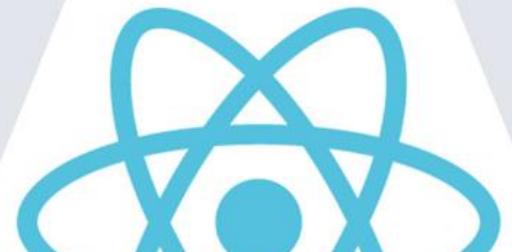


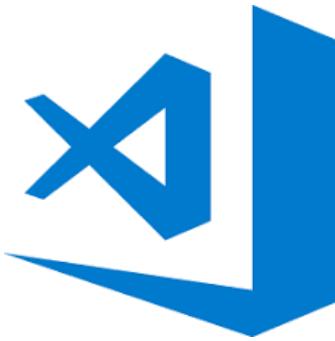
# การเตรียม เครื่องมือ



# | โปรแกรม (Tool and Editor) ที่ใช้บرم

1. Visual Studio Code
2. Node JS 22.x
3. Google Chrome
4. Docker Desktop
5. MongoDB Database

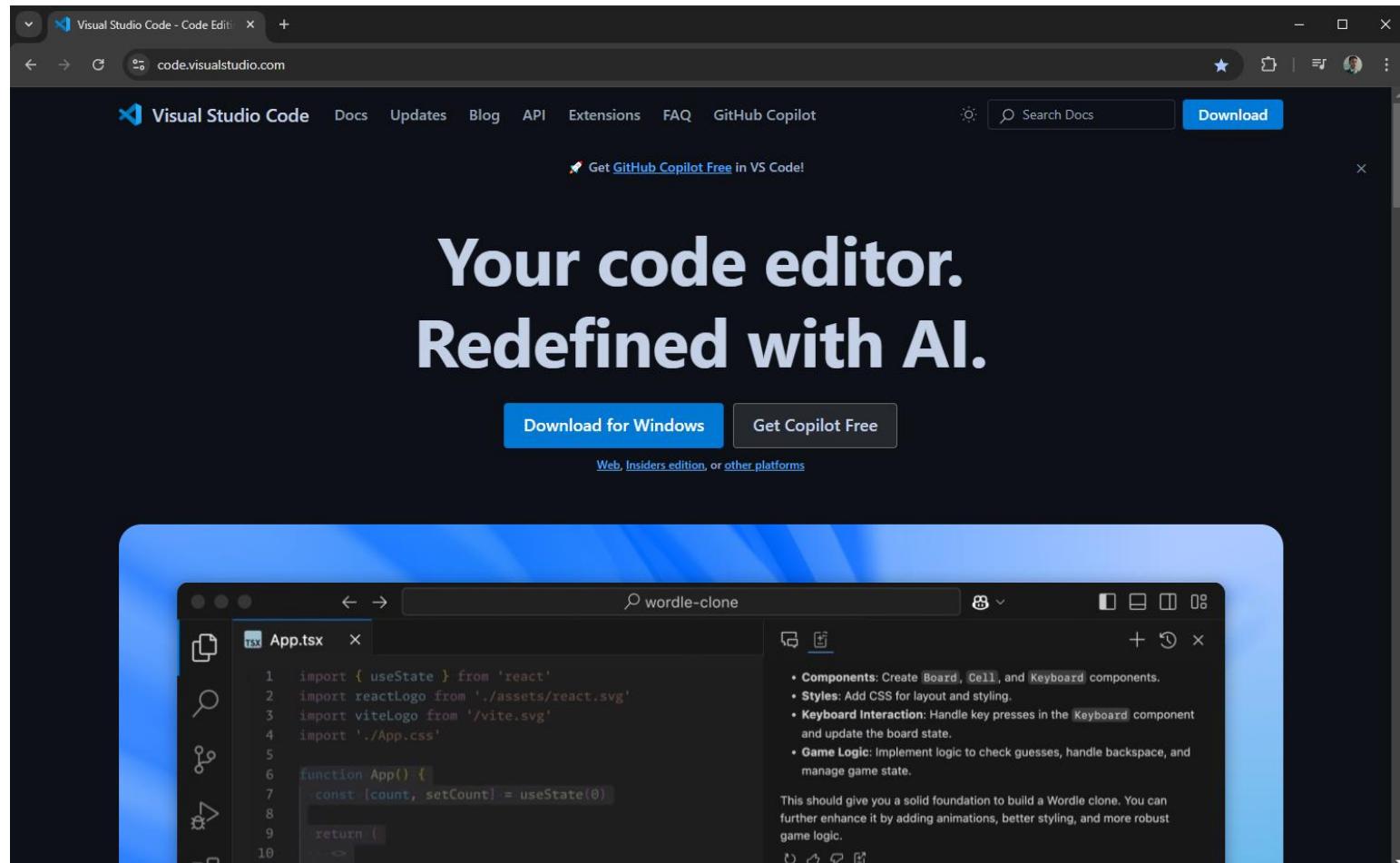




# ติดตั้ง Visual Studio Code

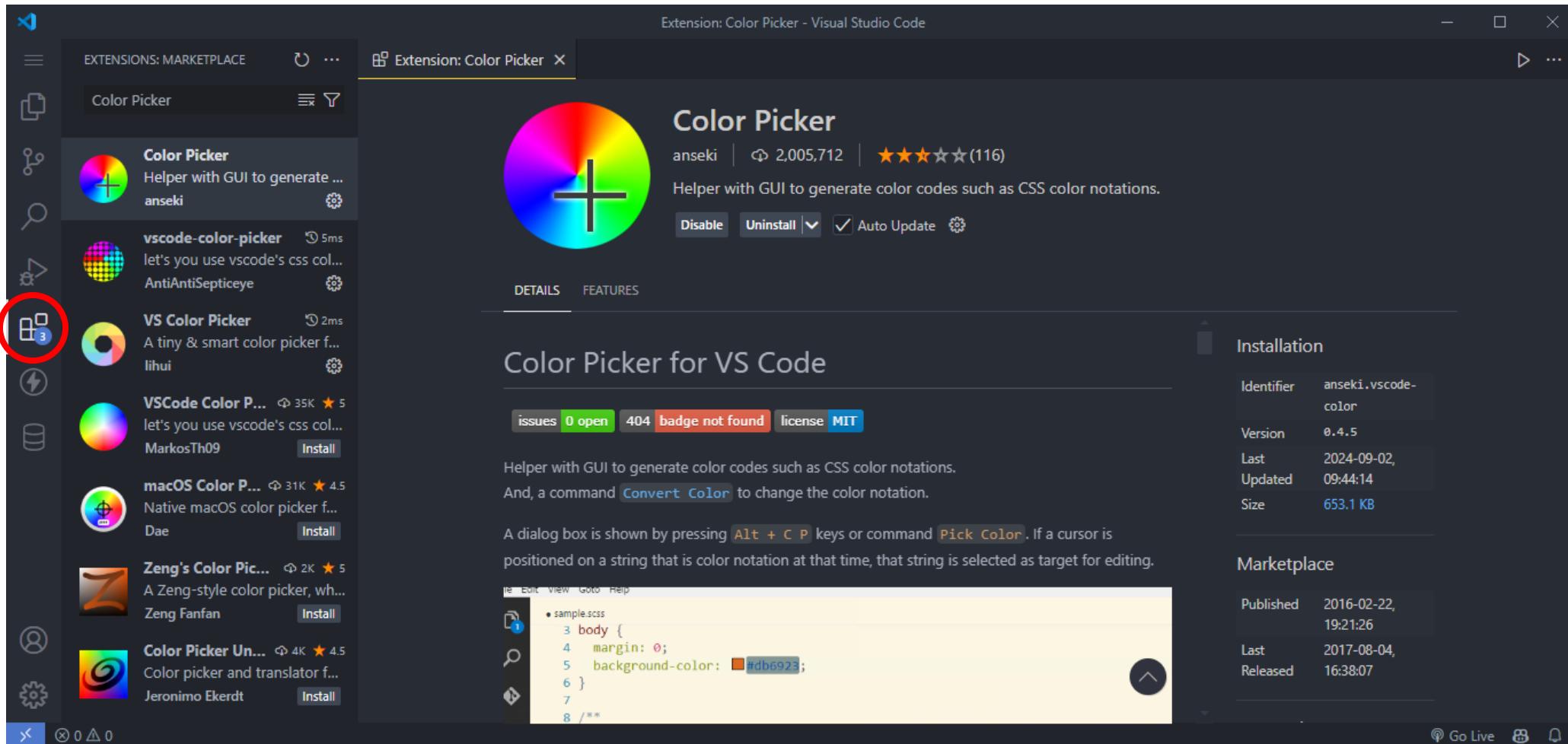


# ติดตั้ง Visual Studio Code พร้อมส่วนเสริมที่จำเป็น



เข้าไปดาวน์โหลด Visual Studio Code ได้ที่ <https://code.visualstudio.com>

# การติดตั้งส่วนเสริม (Extension) ของ Visual Studio Code



# รายการ Extensions ที่แนะนำสำหรับ VS Code

- 1. Color Picker** by anseki
- 2. Material Icon Theme** by Philipp Kief
- 3. ES7 React/Redux/GraphQL/React-Native snippets** by dsznajder
- 4. html to JSX** by Riaz Laskar
- 5. Prettier - Code formatter** by Prettier
- 6. Docker** by Microsoft
- 7. Auto Import** by Sergey Korenuk
- 8. One Dark Pro** by binaryify

# การตั้งค่า (settings.json) ใน VSCode

```
{  
  "emmet.syntaxProfiles": {  
    "javascript": "jsx"  
  },  
  
  "emmet.includeLanguages": {  
    "javascript": "javascriptreact",  
  },  
  
  "emmet.showAbbreviationSuggestions": true,  
  "emmet.showExpandedAbbreviation": "always",  
}
```

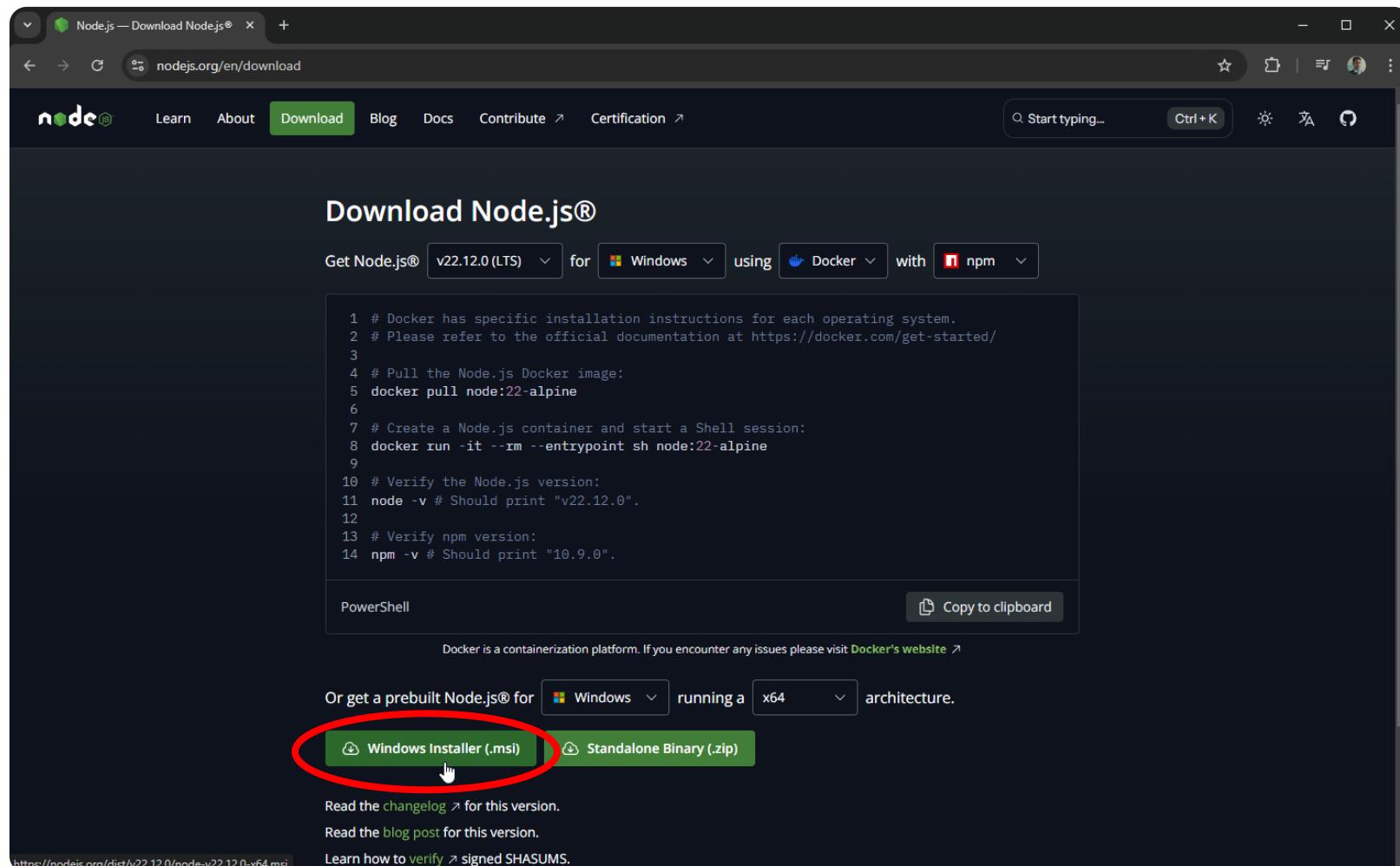


# ติดตั้ง Node JS



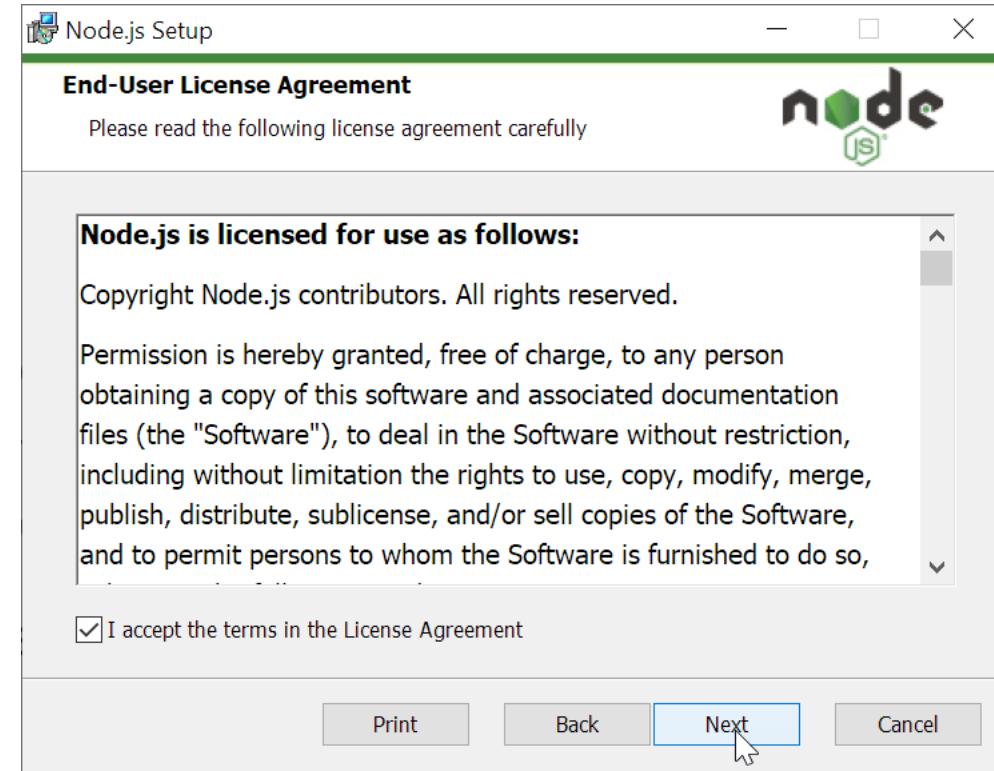
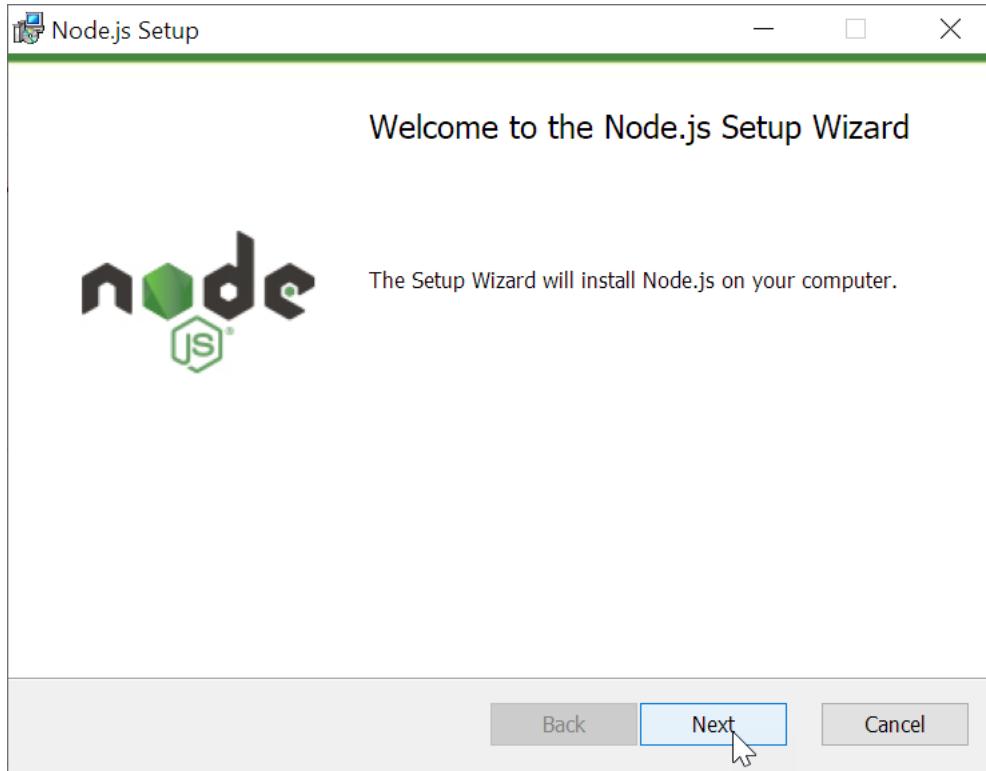
# Download Node.JS V.22.x

<https://nodejs.org/en/download>

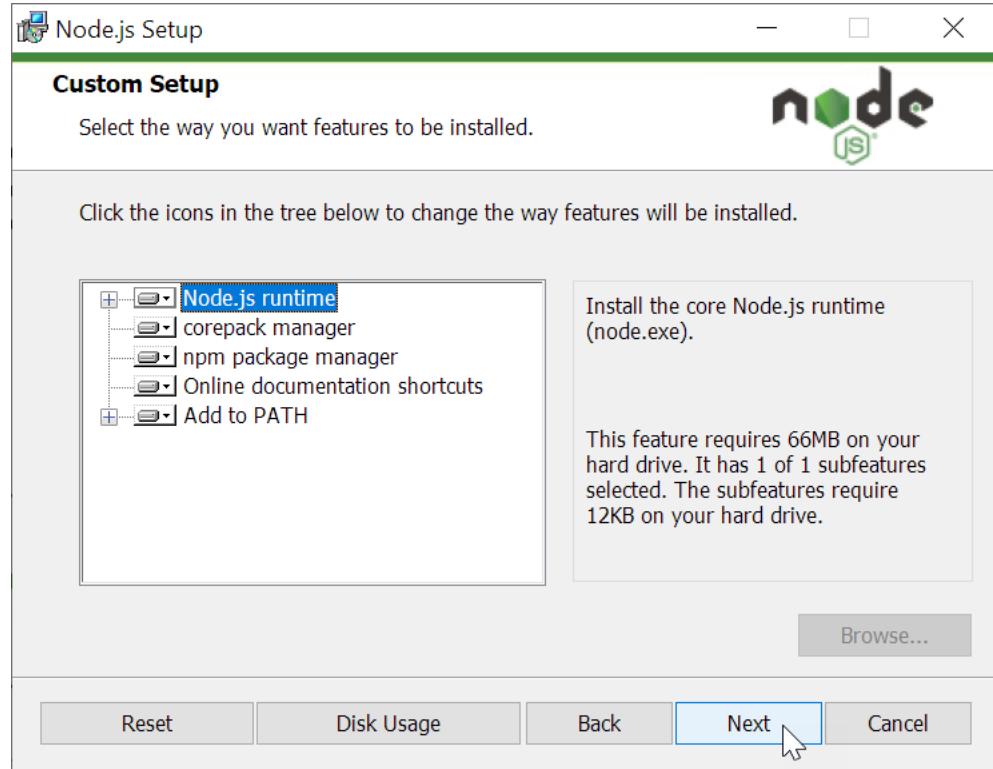
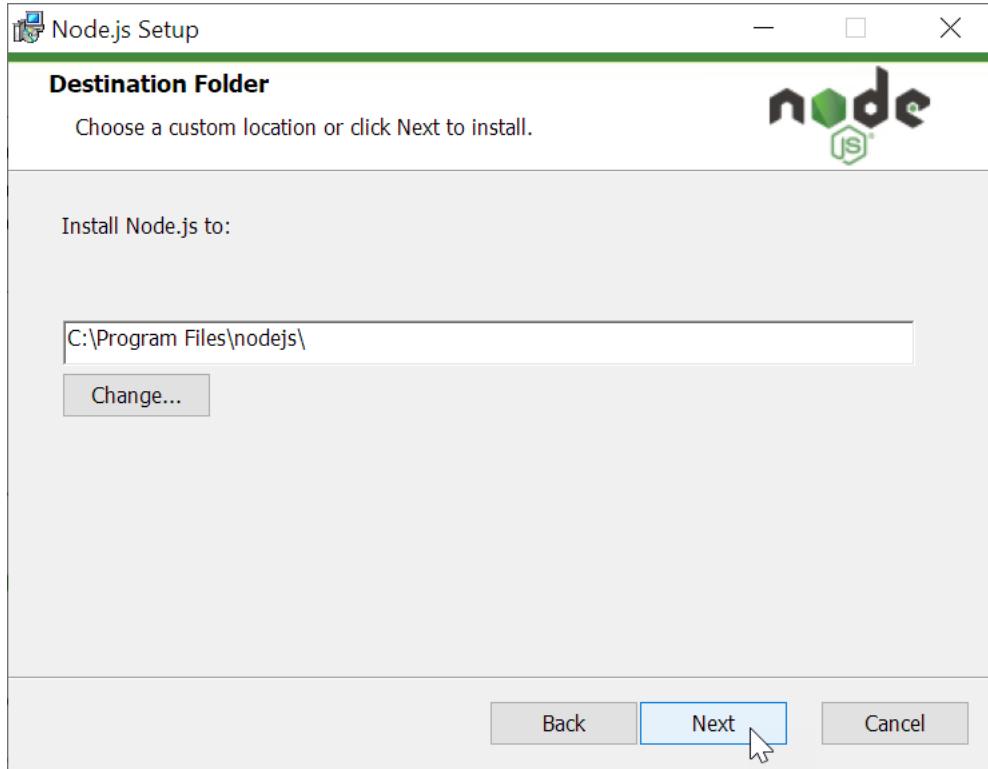


**หมายเหตุ การอัปเดตการอ่านรับตั้งแต่ Node.JS 18 ขึ้นไป**

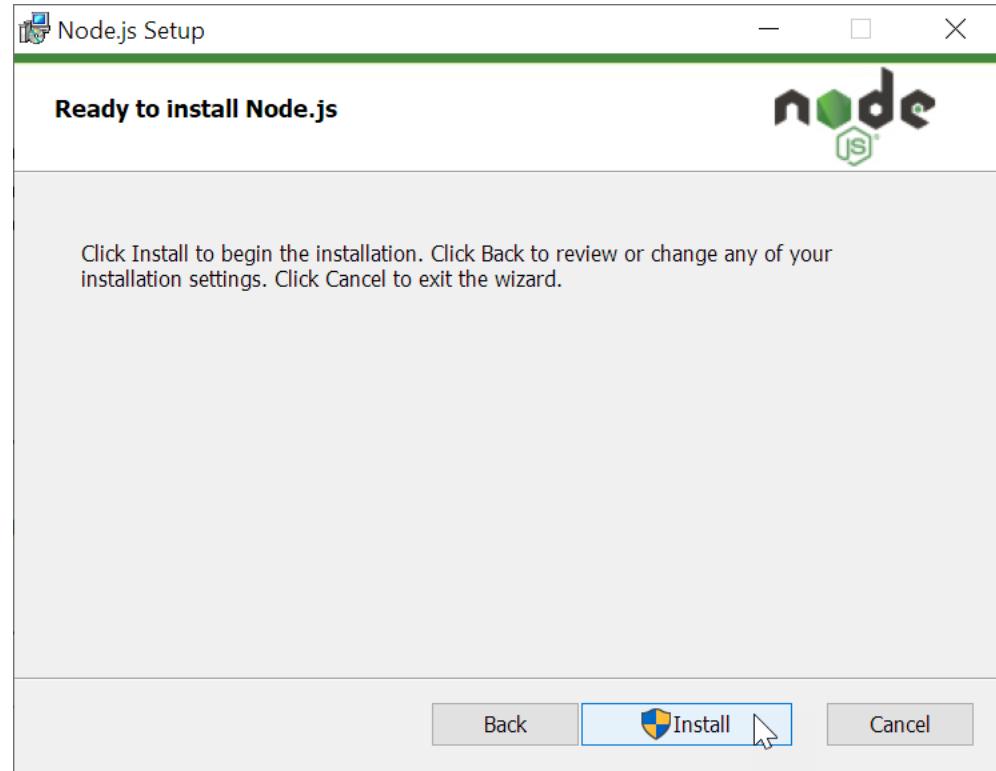
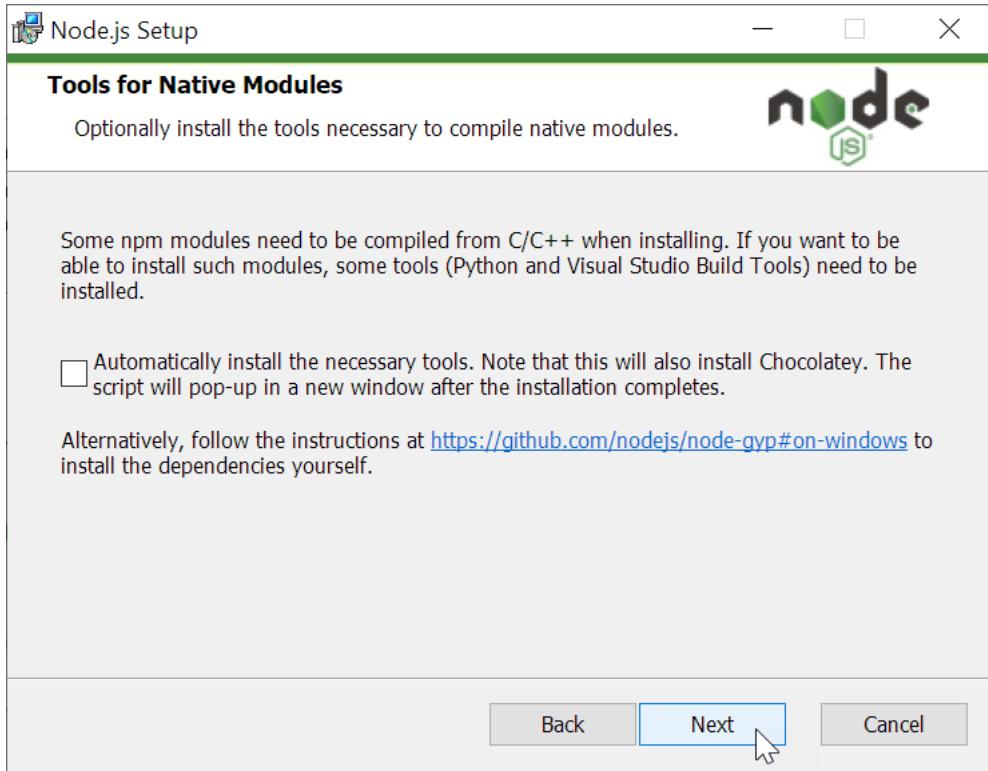
# ติดตั้ง Node.js V.22.x



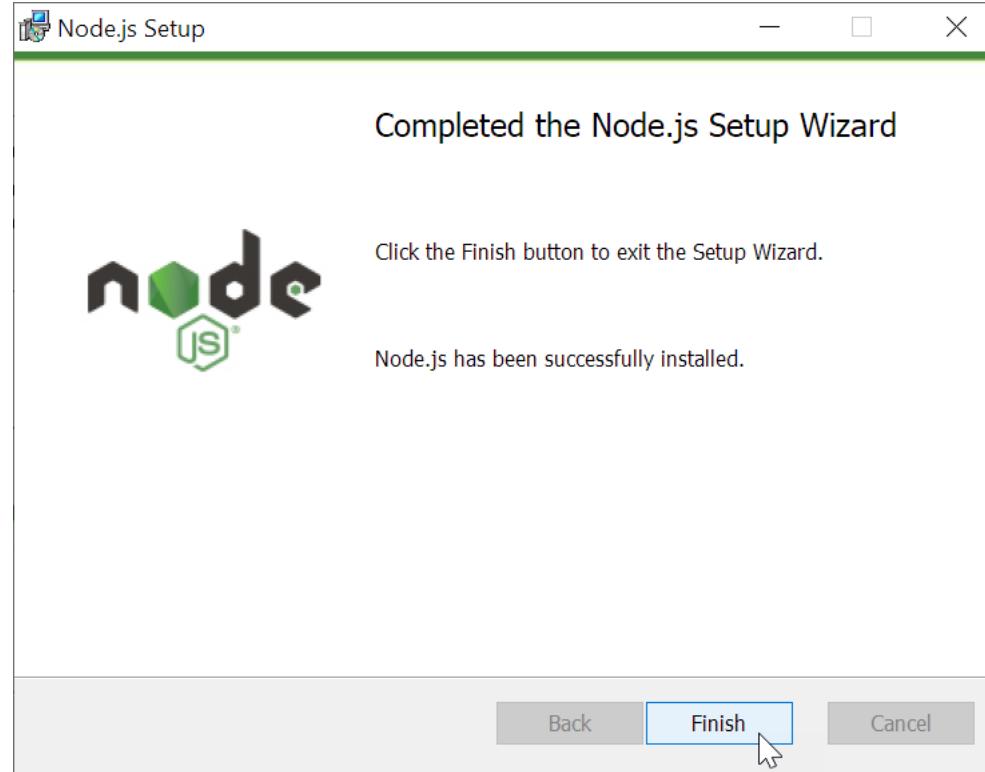
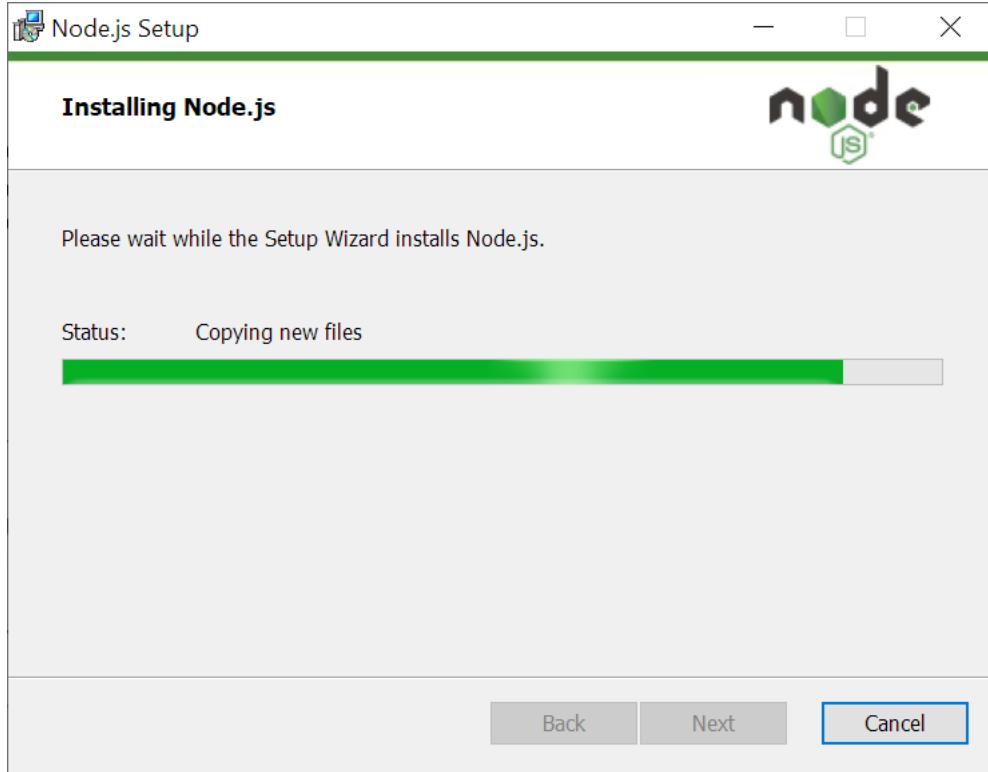
# ติดตั้ง Node.js V.22.x



# ติดตั้ง Node.js V.22.x



# ติดตั้ง Node.js V.22.x



# nodejs ဆုတေသန

```
node -v
x + v
C:\Users\samit>node -v
v22.1.0
C:\Users\samit>
```

```
npx -v
x + v
C:\Users\samit>npx -v
10.7.0
C:\Users\samit>
```

```
npm -v
x + v
C:\Users\samit>npm -v
10.7.0
C:\Users\samit>
```



chrome

# ติดตั้งเครื่องมือบน Google Chrome สำหรับใช้พัฒนา React



React Developer Tools - Chrome

chrome.google.com/webstore/detail/react-developer-tools/fmkadmapgofadopljbjfkapdkoienihi

chrome เว็บสโตร์ samitkoyom@gmail.com

หน้าแรก > ส่วนขยาย > React Developer Tools

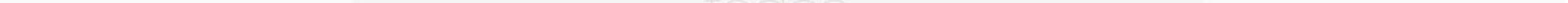
**React Developer Tools**

เขียนโดย: Facebook

★★★★★ 1,304 | เครื่องมือสำหรับนักพัฒนาซอฟต์แวร์ | ผู้ใช้ 2,000,000+ ราย

ดาวน์โหลดจาก Chrome

ภาพรวม ความเห็น สนับสนุน รายการที่เกี่ยวข้อง



Redux DevTools - Chrome เร็วๆนี้ x +

chrome.google.com/webstore/detail/redux-devtools/lmhkpmbekcpmknkloieibfkpmffibljd

chrome เว็บสโตร์ samitkoyom@gmail.com

หน้าแรก > ส่วนขยาย > Redux DevTools

Redux DevTools

เขียนโดย: remotedevio

★★★★★ 532 | เครื่องมือสำหรับนักพัฒนาซอฟต์แวร์ | ผู้ใช้ 1,000,000+ ราย

ปิดออกจาก Chrome

ภาพรวม ความเห็น สนับสนุน รายการที่เกี่ยวข้อง

Redux DevTools

state todos

todos[0], todos[1], todos[2]

Log monitor

REDINIT

COMPLETE\_TODO

ADD\_TODO

ADD\_TODO

todos[0]

completed: false => true



# ติดตั้ง Docker Desktop



# System Requirements Windows



**Docker Engine 20.10.9**

**Docker Desktop 4.10**



- Windows 10 64-bit: Pro 2004 (build 19041) or higher, or Enterprise or Education 1909 (build 18363) or higher.
- For Windows 10 Home, see System requirements for WSL 2 backend.
- Hyper-V and Containers Windows features must be enabled.
- The following hardware prerequisites are required to successfully run Client Hyper-V on Windows 10:
  - 64 bit processor with Second Level Address Translation (SLAT)
  - 4GB system RAM
  - BIOS-level hardware virtualization support must be enabled in the BIOS settings

<https://docs.docker.com/desktop/windows/install/>

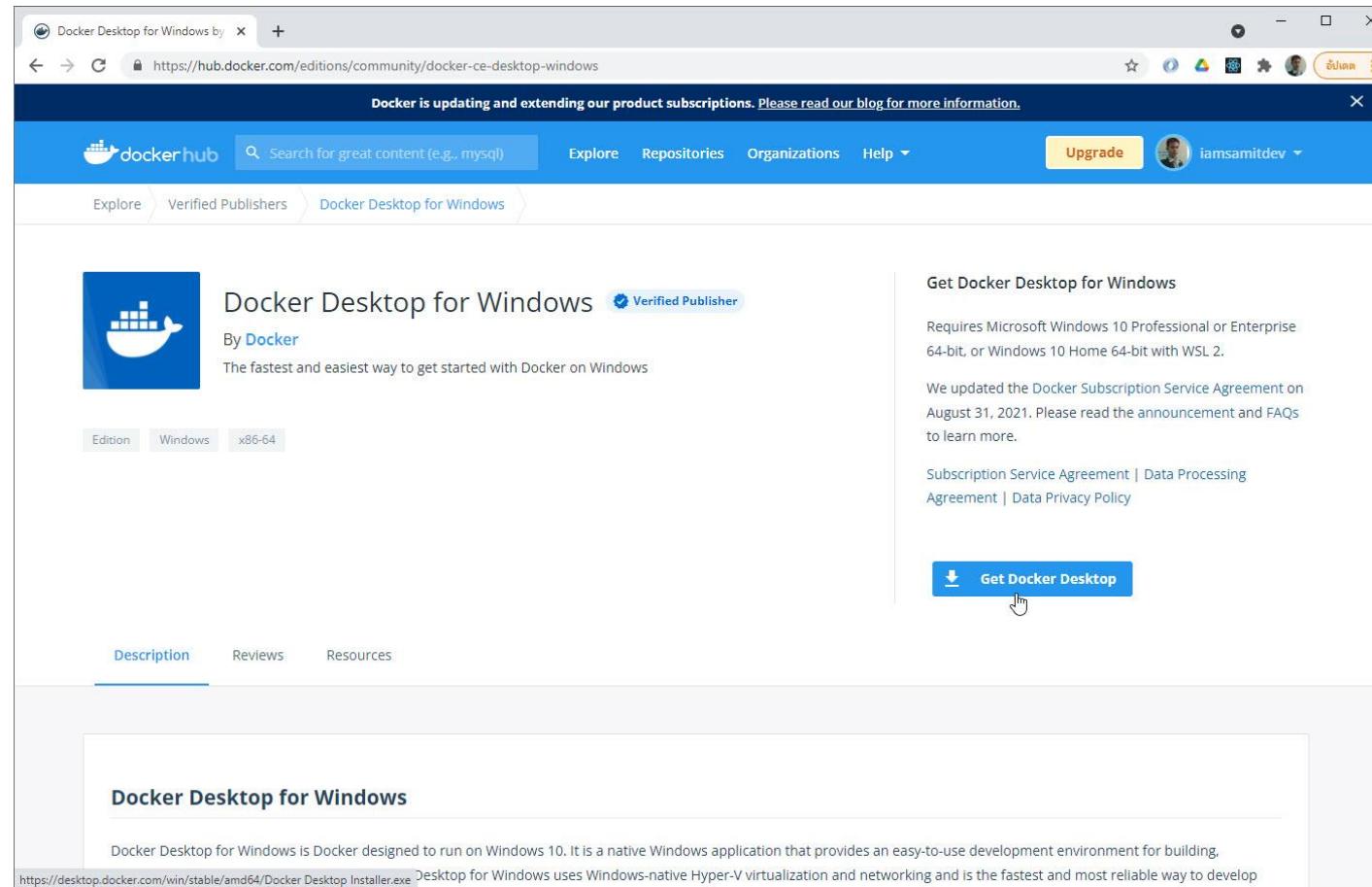
# วิดีโอบนเน็ตขั้นตอนการติดตั้ง Docker Desktop อย่างละเอียด

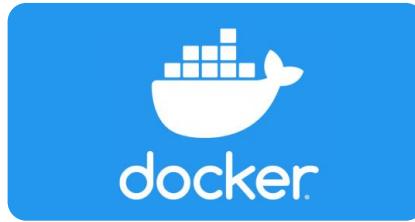


<https://www.youtube.com/watch?v=Uo5rkcZFHC'A>

# Download Docker Desktop for Windows

<https://hub.docker.com/editions/community/docker-ce-desktop-windows>





# ติดตั้ง Docker Desktop



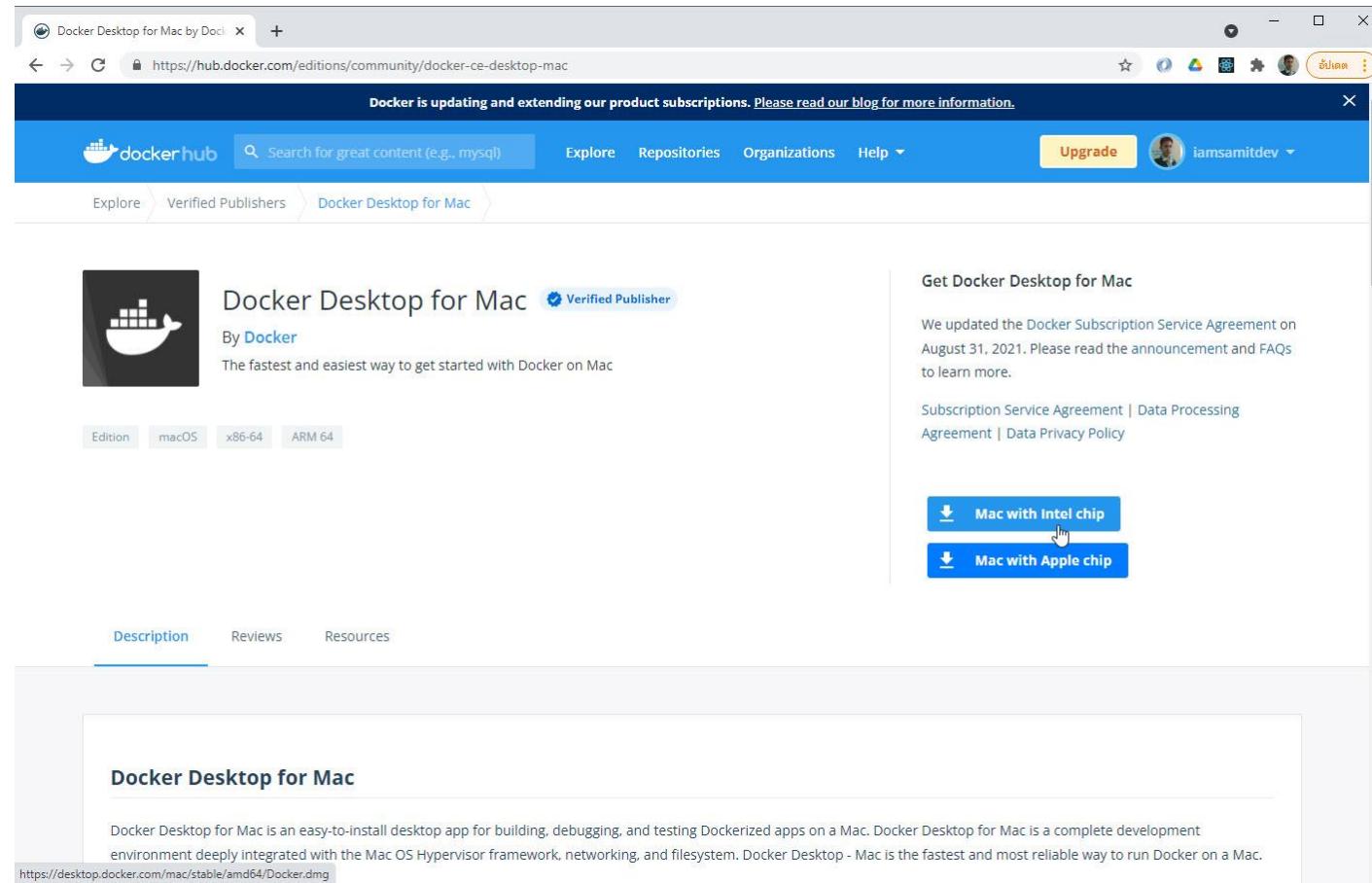
# วิดีโอแนะนำขั้นตอนการติดตั้ง Docker Desktop อย่างละเอียด



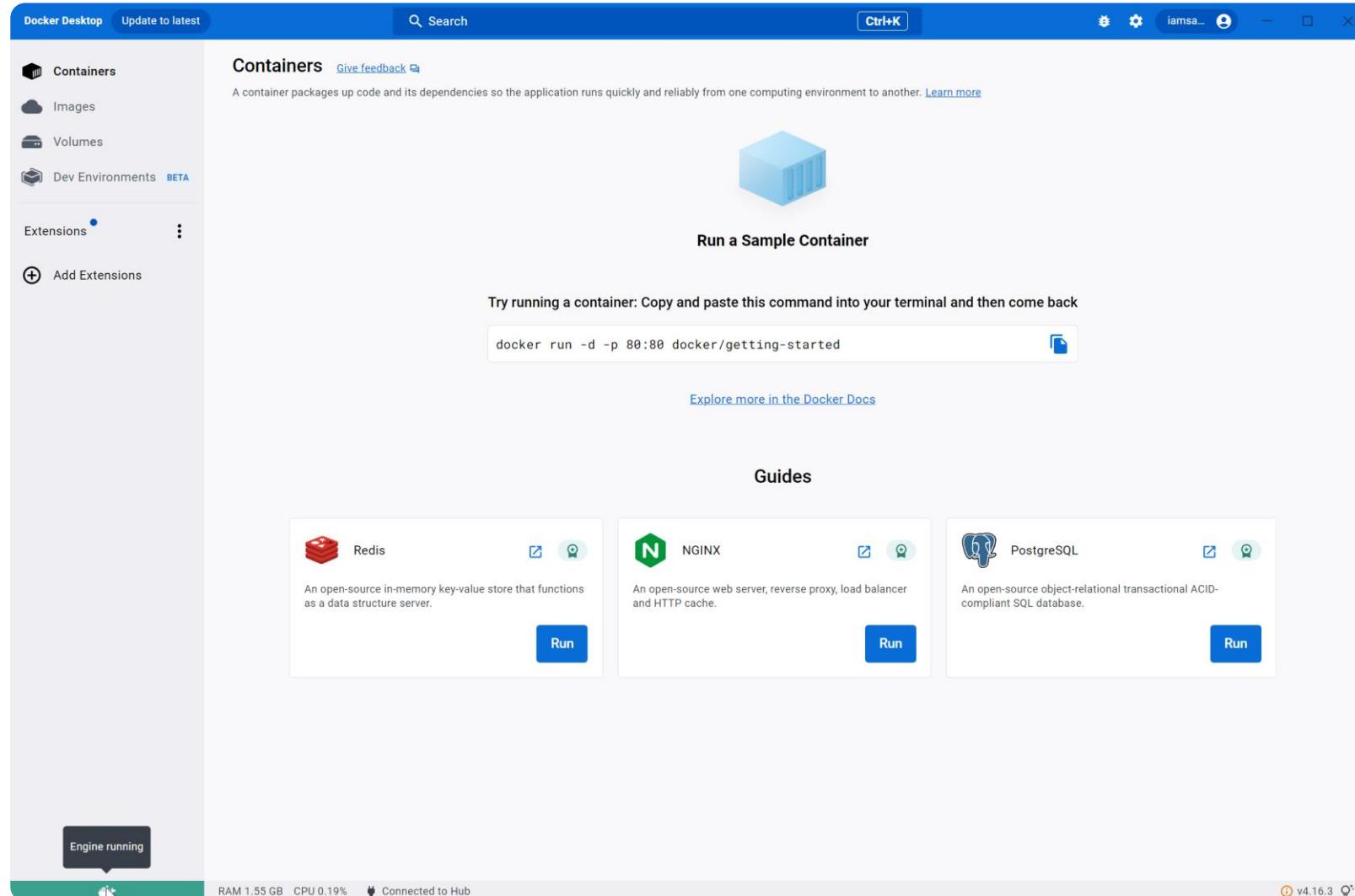
<https://www.youtube.com/watch?v=Lbr3FnsfAAU>

# Download Docker Desktop for MacOS

<https://hub.docker.com/editions/community/docker-ce-desktop-mac>



# ตัวอย่าง Docker Desktop หลังจากติดตั้งเรียบร้อยแล้ว



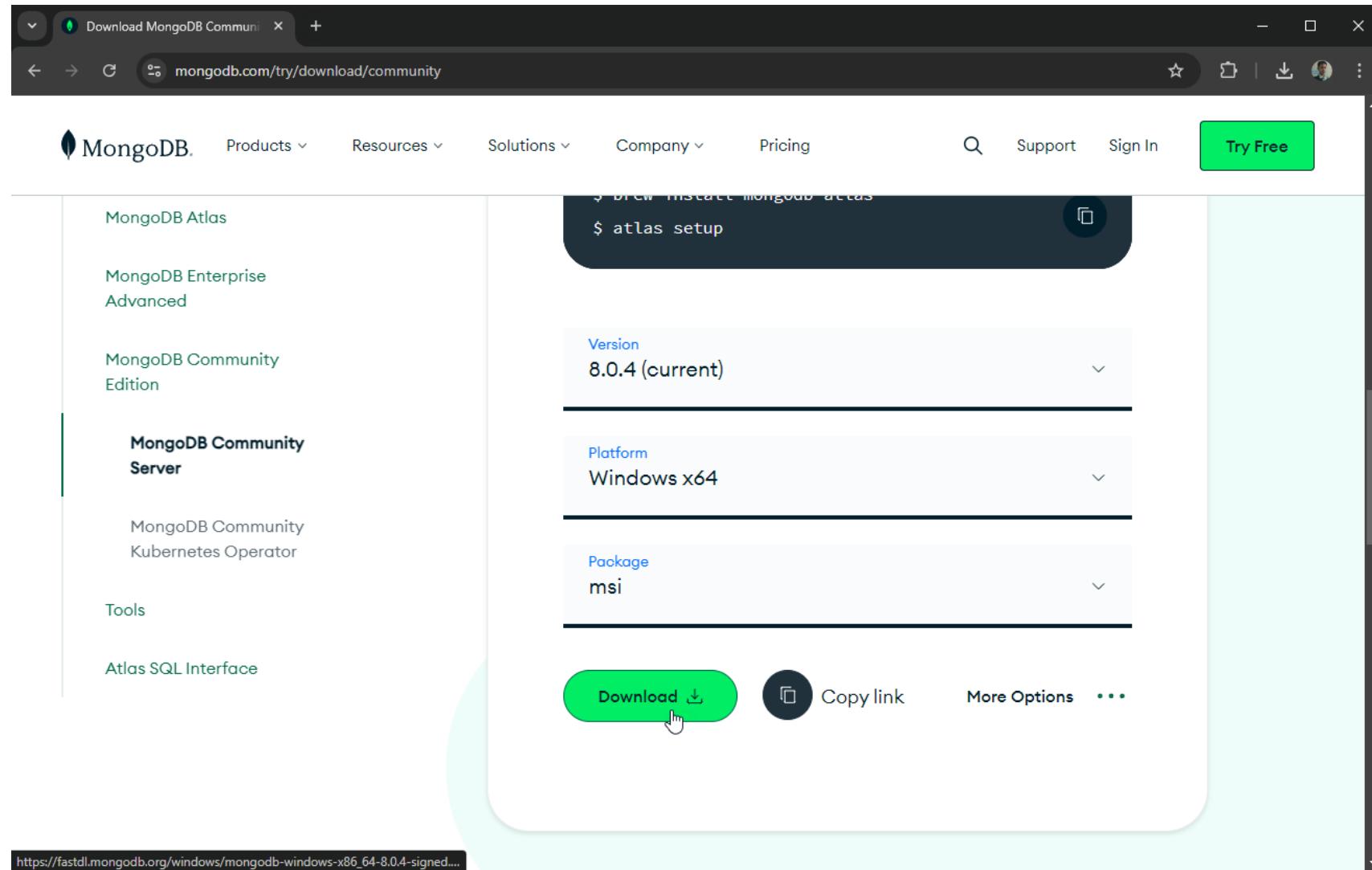


# ติดตั้ง MongoDB

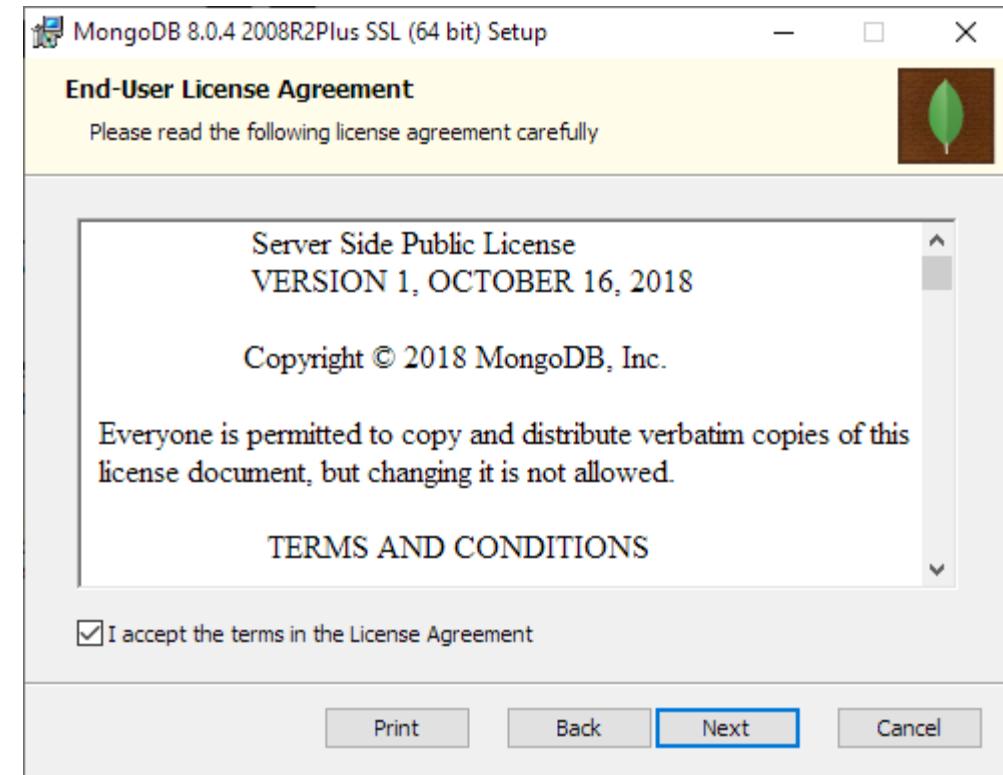
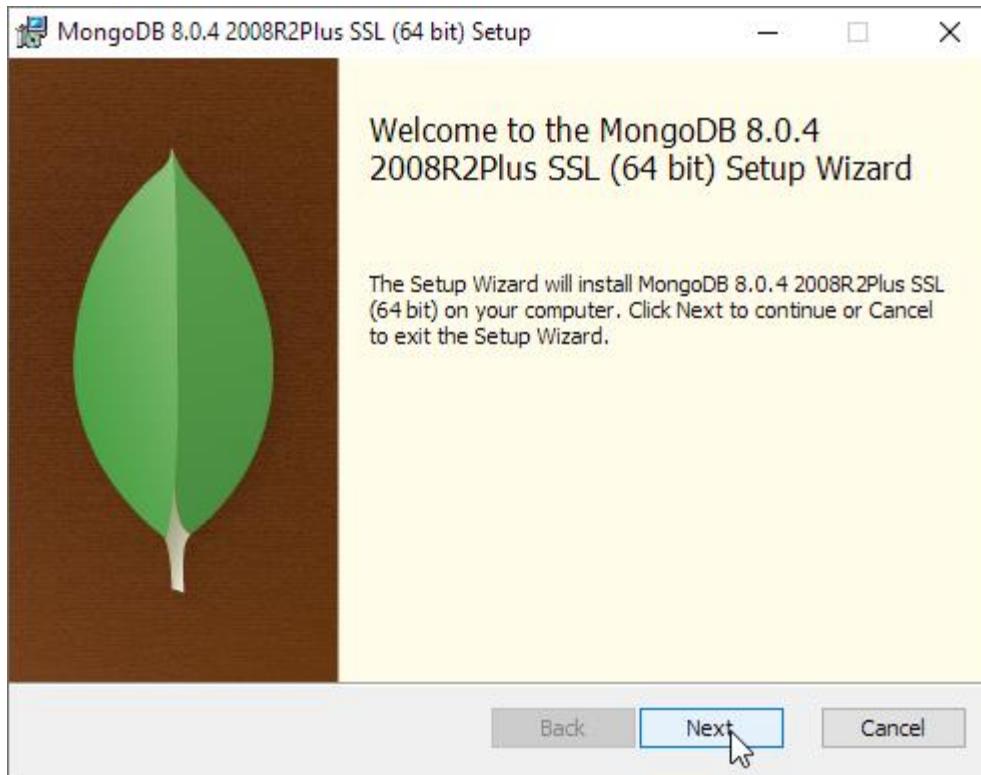


# สามารถเข้าไปดาวน์โหลดได้ที่

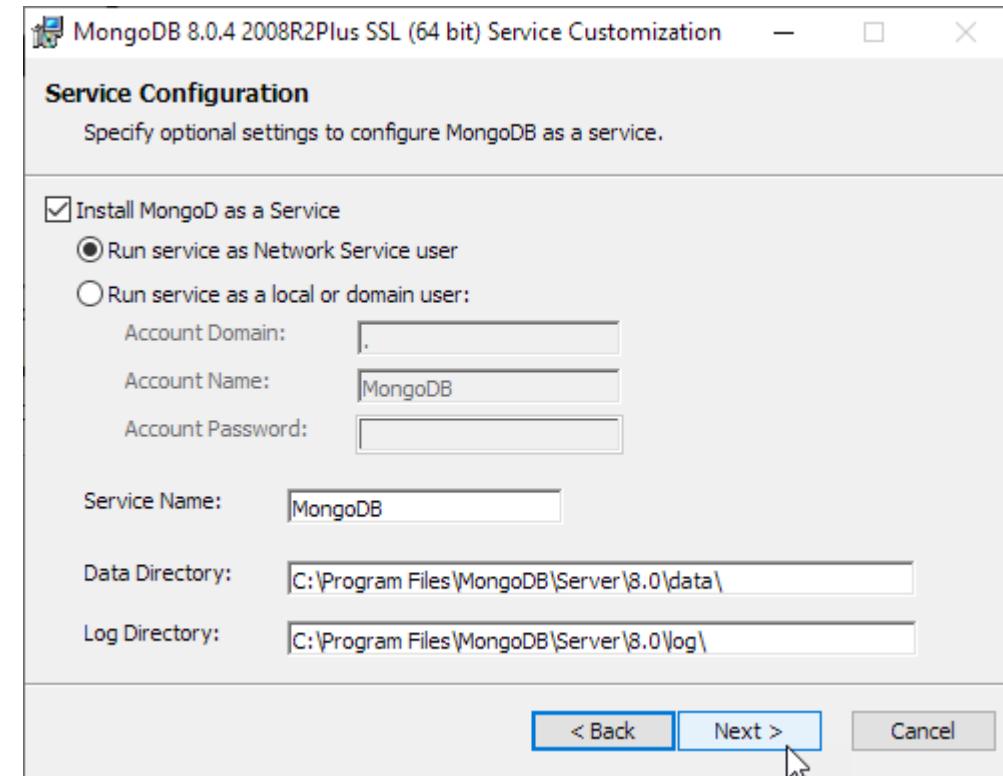
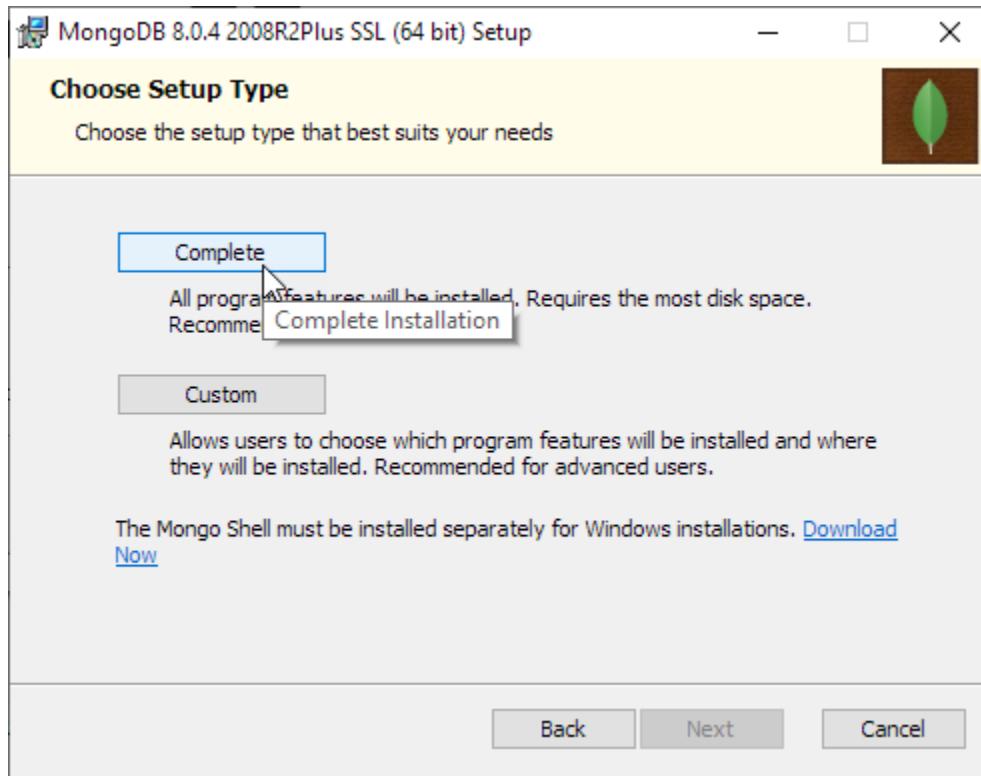
<https://www.mongodb.com/try/download/community>



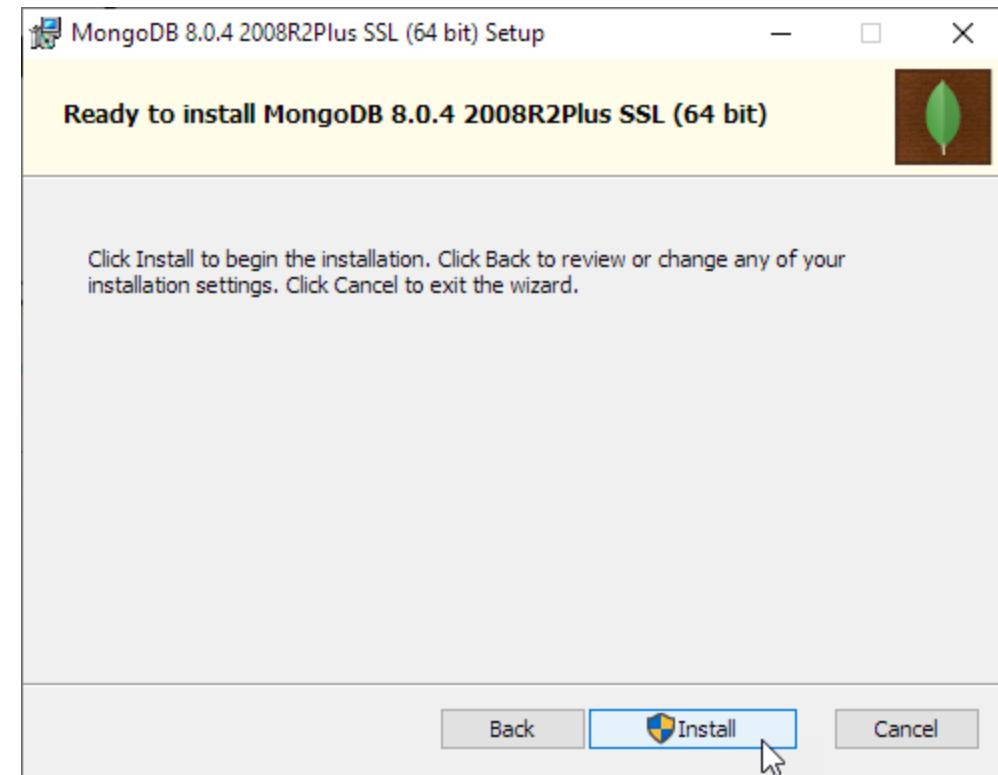
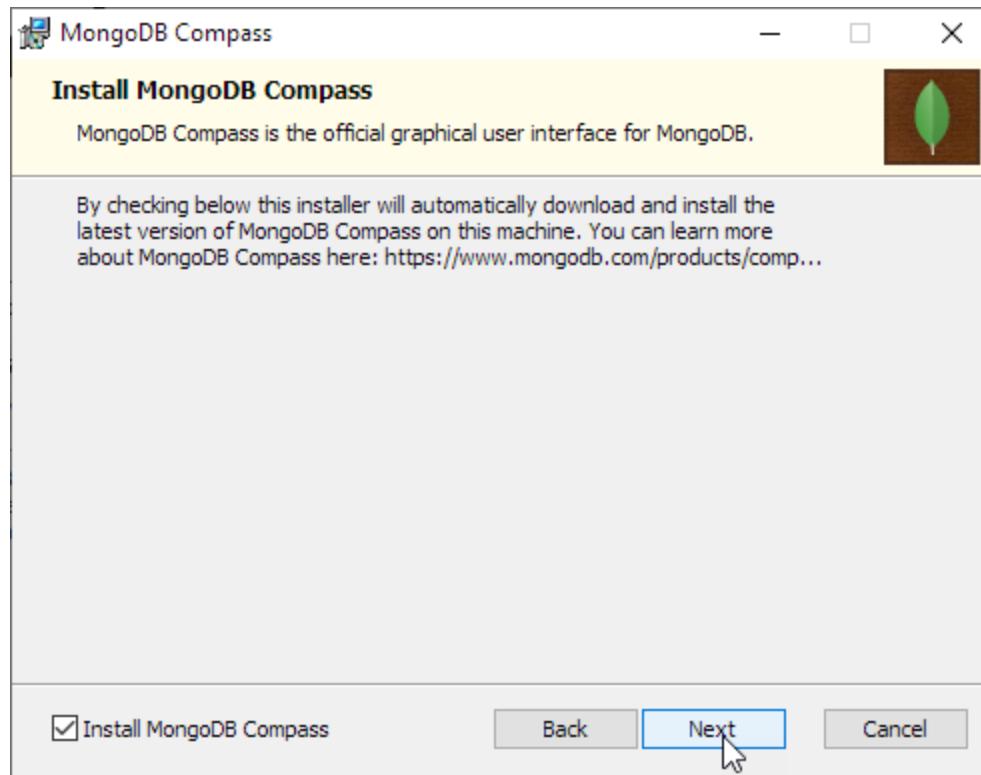
# ติดตั้ง MongoDB



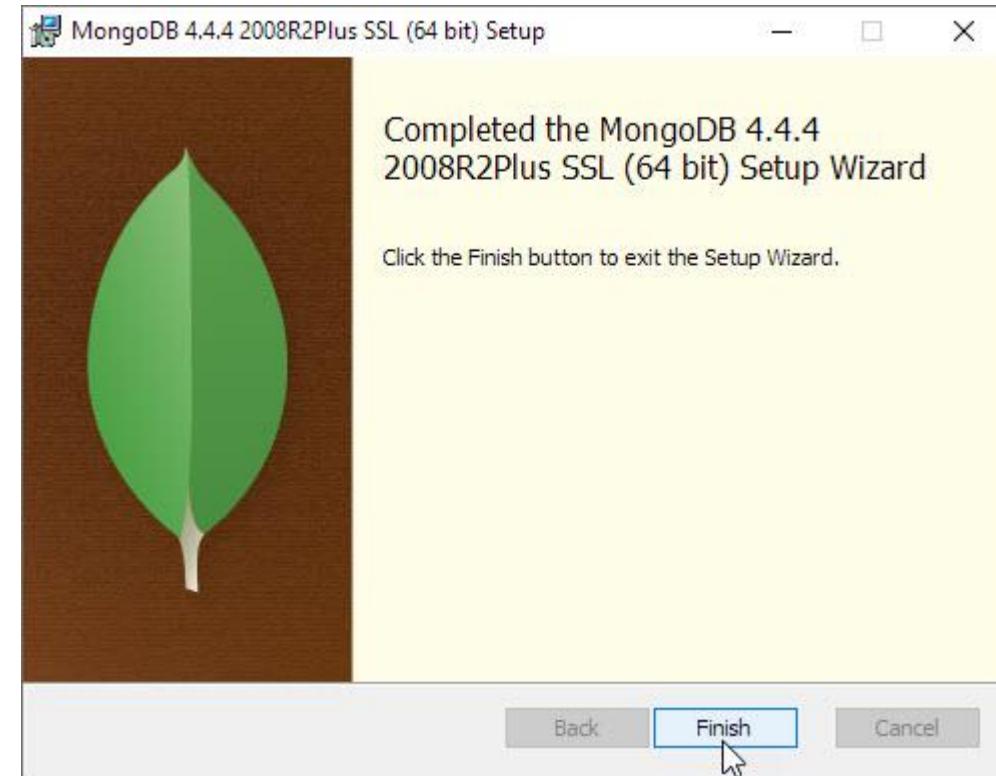
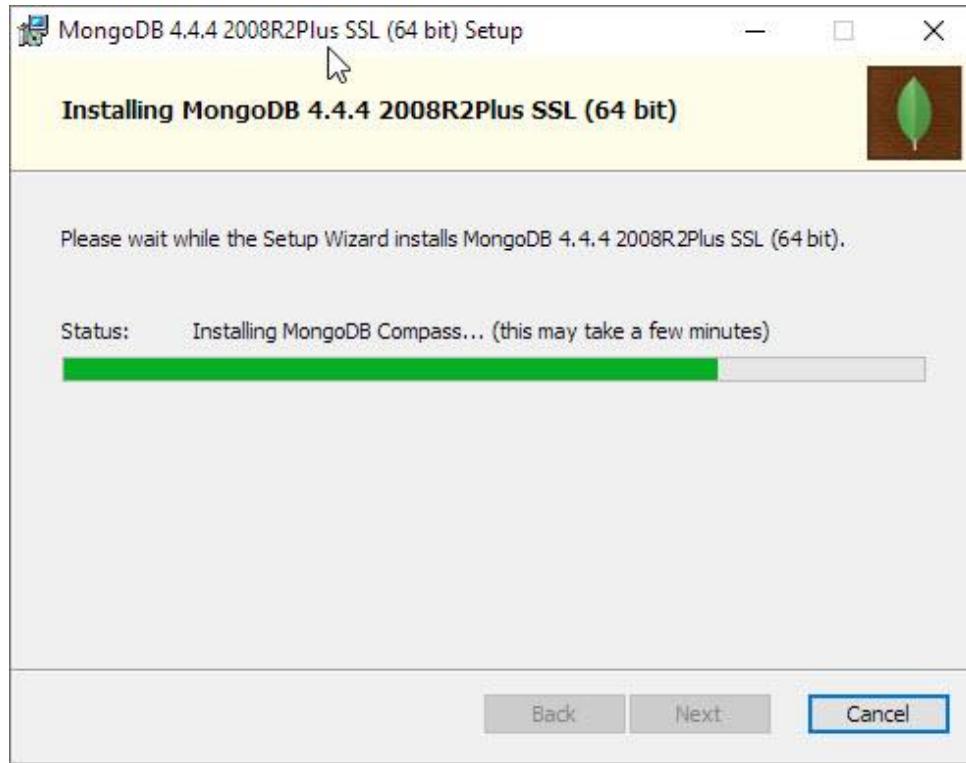
# ติดตั้ง MongoDB



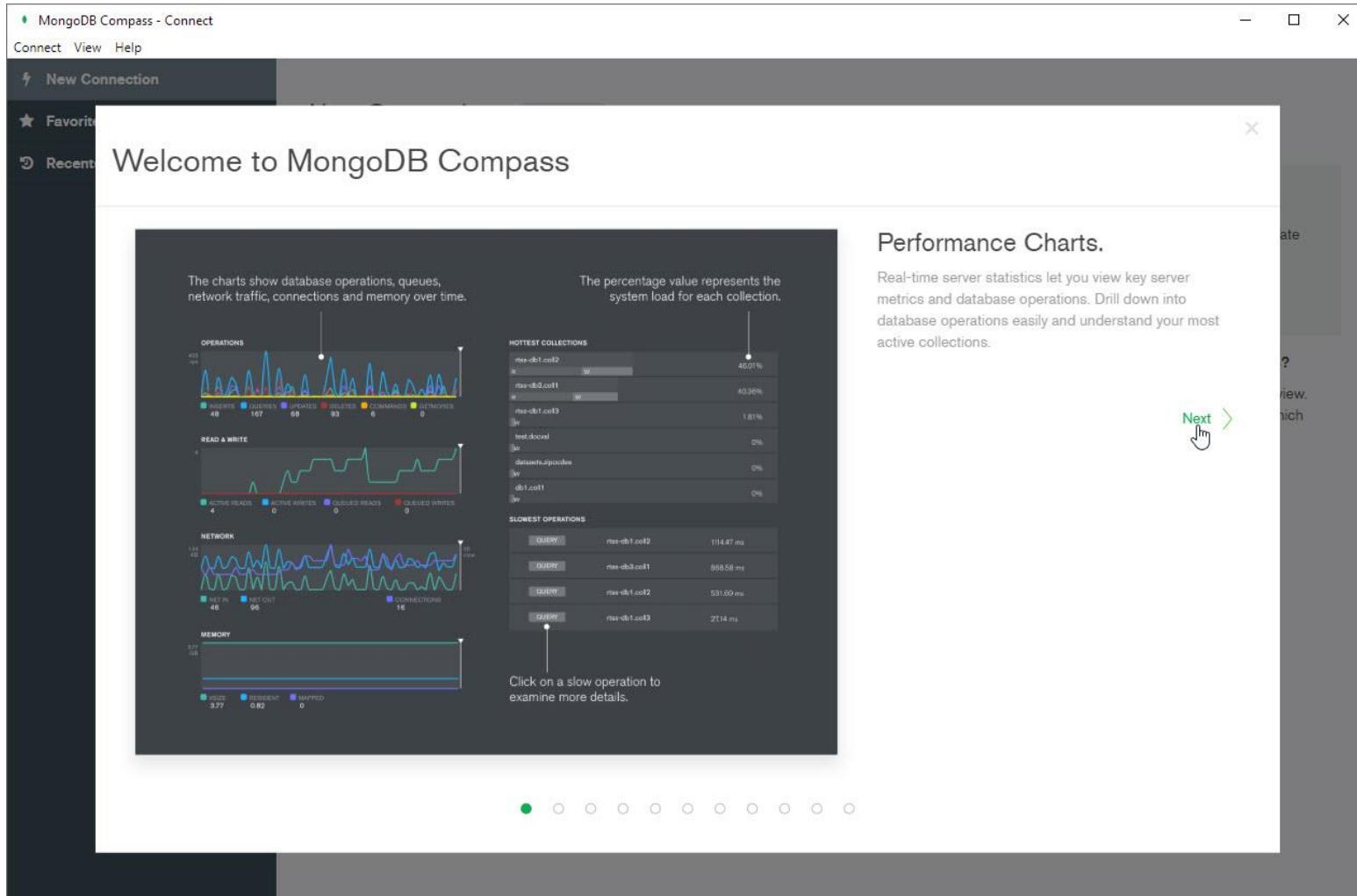
# ติดตั้ง MongoDB



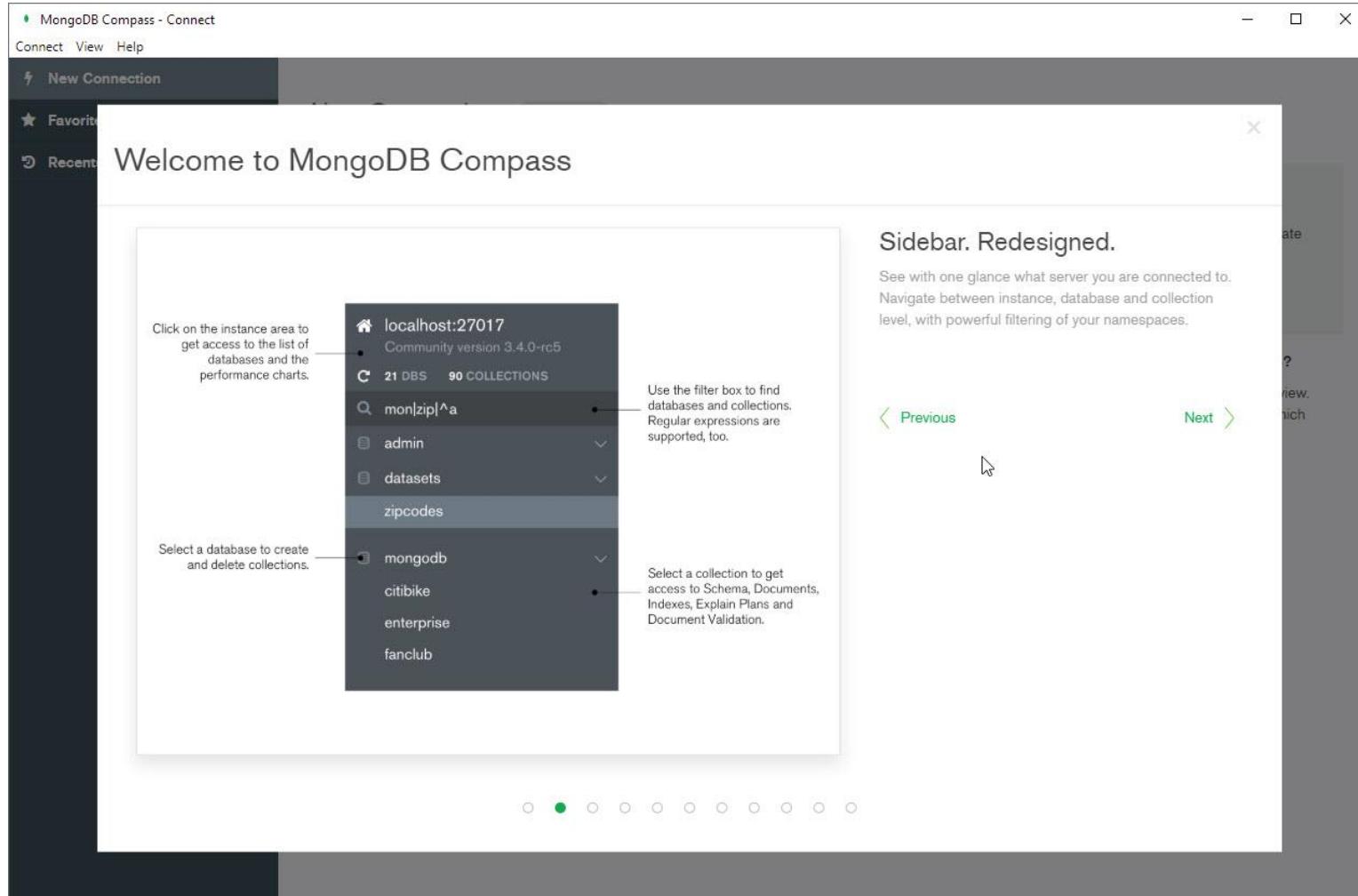
# ติดตั้ง MongoDB



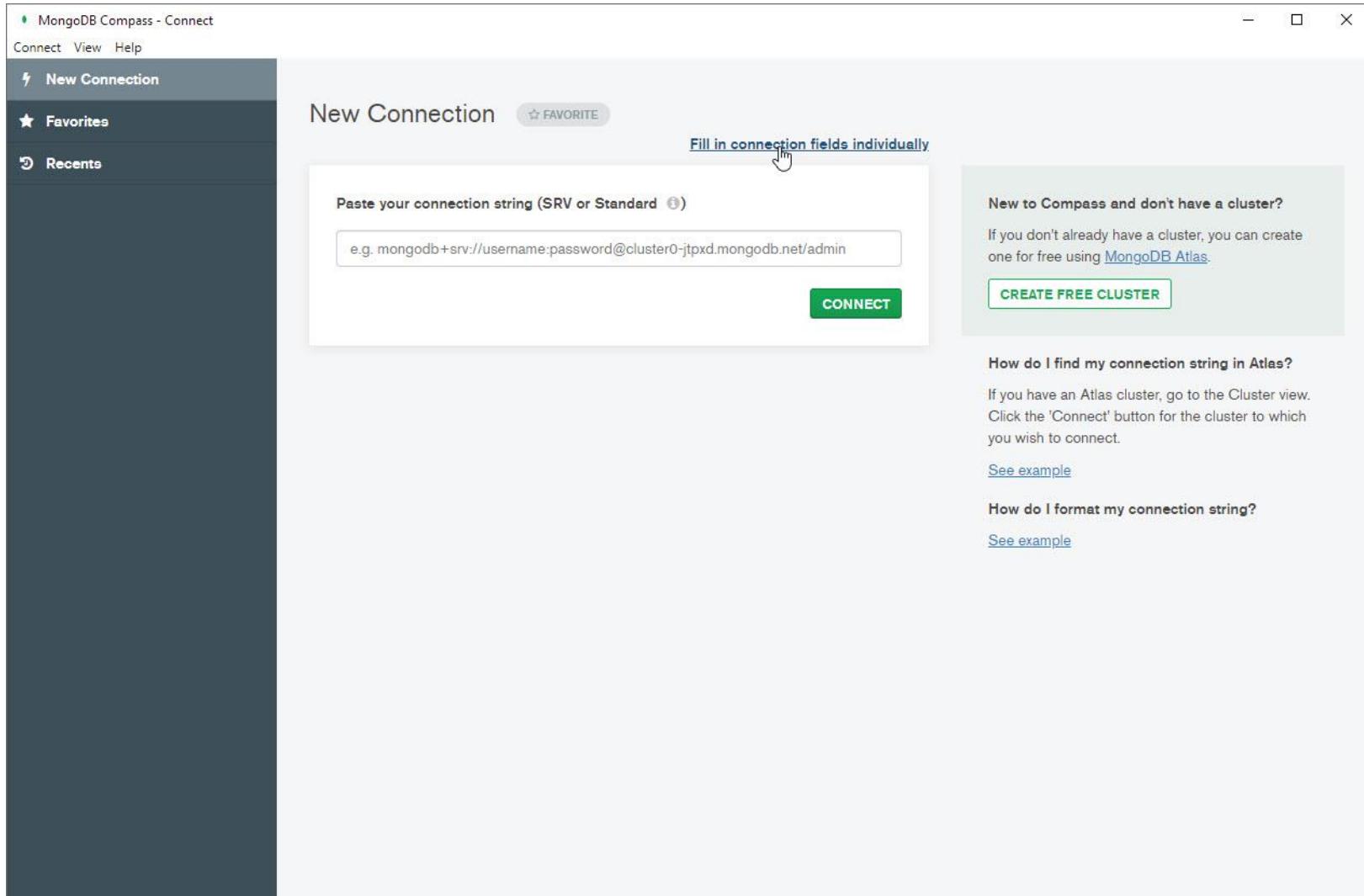
# ຕິດຕັ້ງ MongoDB



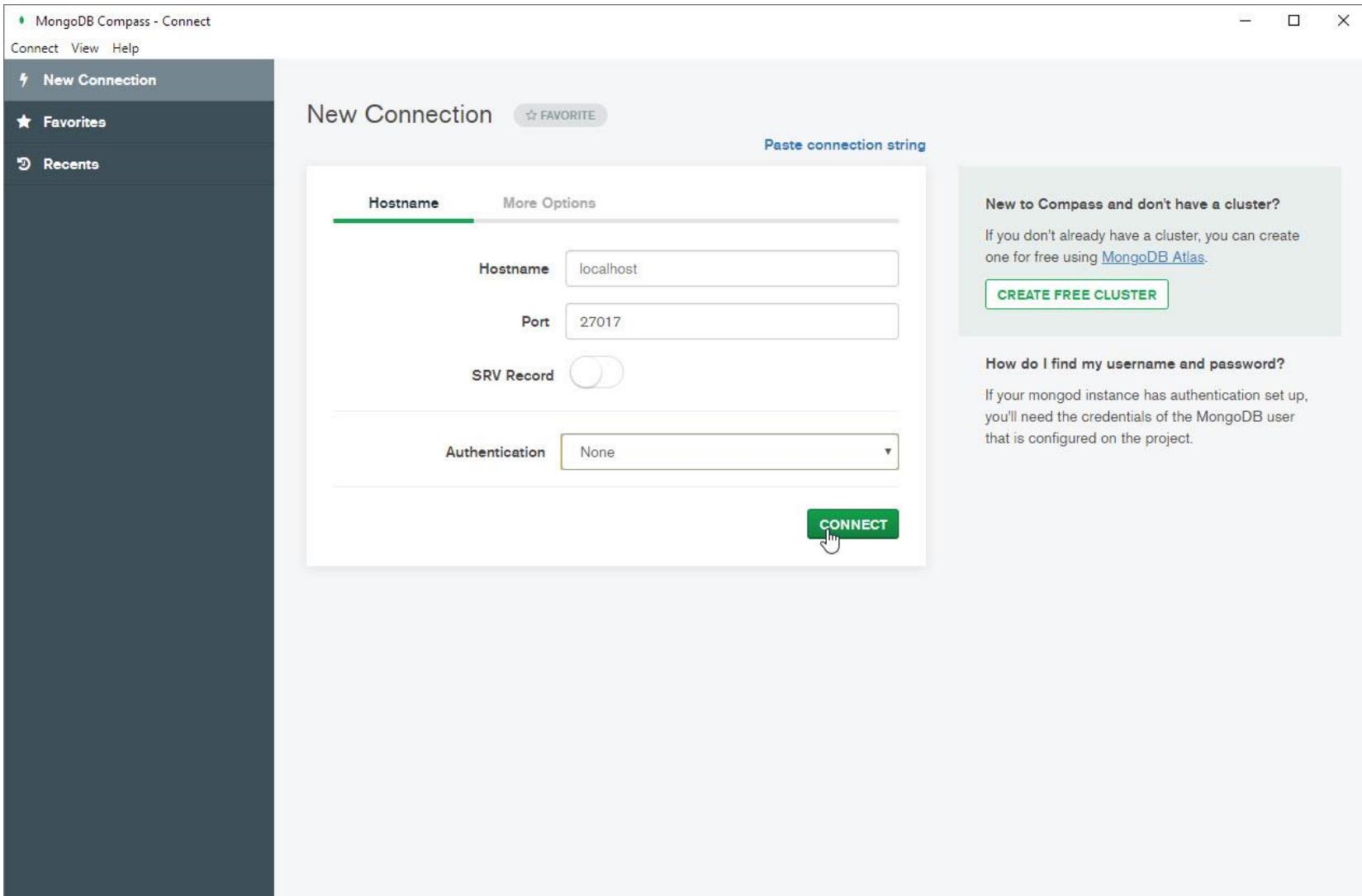
# ติดตั้ง MongoDB



# ຕິດຕັ້ງ MongoDB



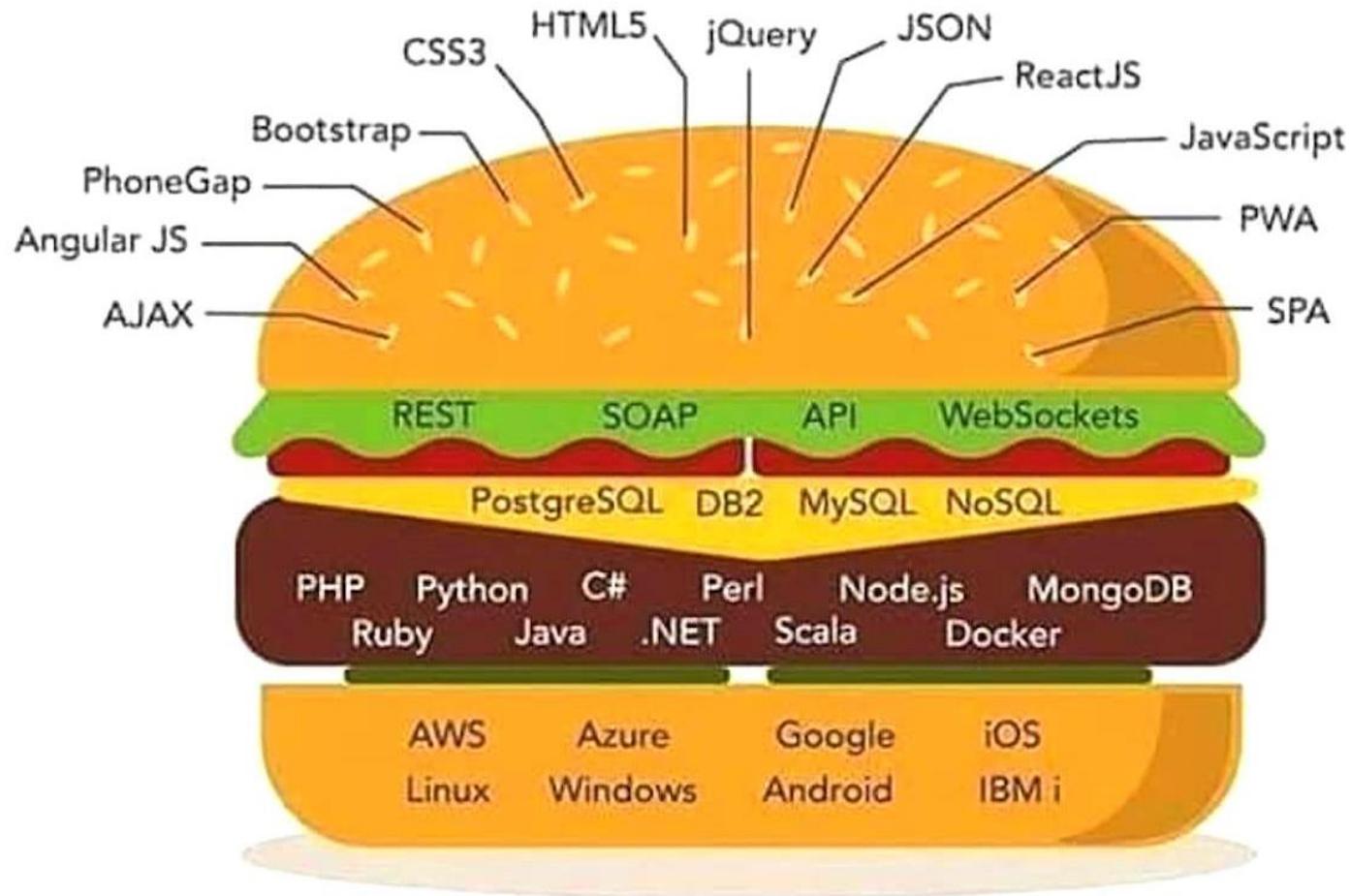
# ติดตั้ง MongoDB

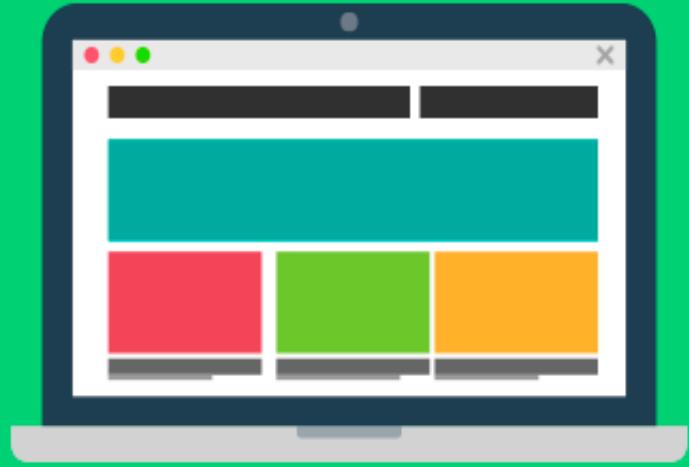


The background image shows a paved road curving through a landscape of green fields and yellow grass, bordered by small orange markers. In the distance, a range of mountains with dark, rugged peaks rises against a light blue sky with scattered clouds.

# Web Developer Roadmap 2025

# “The Full-Stack”

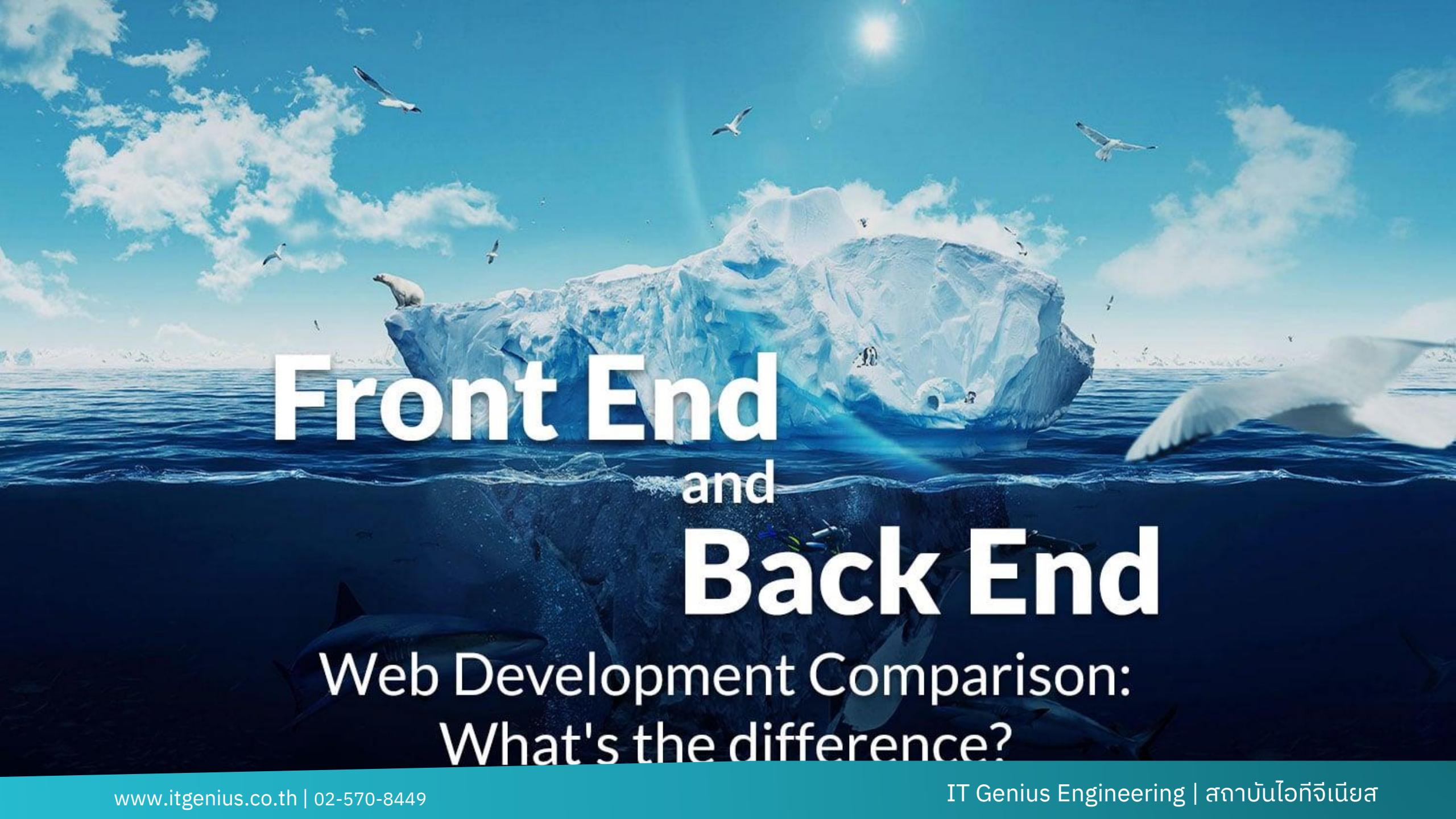




**FRONTEND**



**BACKEND**



# Front End and Back End

## Web Development Comparison: What's the difference?

# { FRONT-END }

CSS

HTML

Javascript



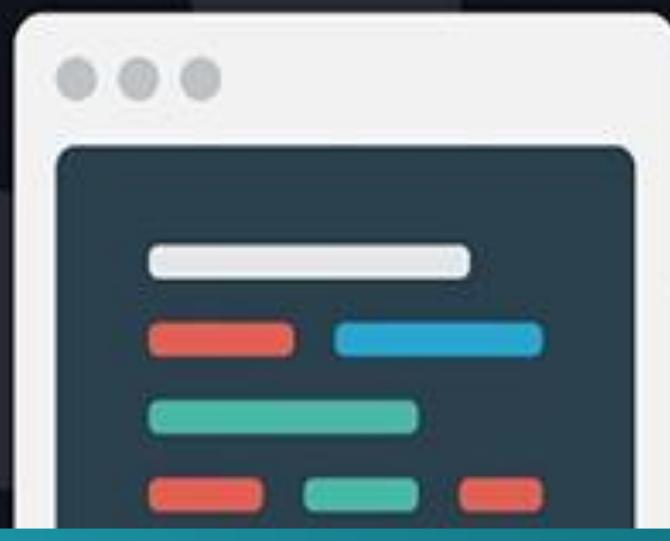
# < BACK-END >

PHP

.NET

ExpressionEngine

VS



The screenshot shows a web-based roadmap titled "Full Stack Developer". At the top, there are navigation links: "All Roadmaps", "Schedule Learning Time" (with a bookmark icon), "Download" (with a download icon), and "Share" (with a share icon). Below the title, a subtitle reads "Step by step guide to becoming a modern full stack developer in 2025". There are three tabs: "Roadmap" (selected), "Projects", and "soon". On the right, there's a "Suggest Changes" button. Below the tabs, a progress bar shows "0% DONE" and "0 of 37 Done". A dropdown menu is open, showing the first item: "What is Full Stack Development?". To the left, a vertical blue dotted line is labeled "Full Stack". The main content area contains two boxes. The left box contains text about existing full-stack developers and links to "Frontend", "Backend", and "DevOps". The right box contains text about finding detailed versions and links to "roadmap.sh". At the bottom, there are two icons: a yellow square for "Key topics to learn" and a black square for "Project ideas and suggestions".

← All Roadmaps

Schedule Learning Time

Download

Share

# Full Stack Developer

Step by step guide to becoming a modern full stack developer in 2025

Roadmap Projects soon

0% DONE 0 of 37 Done

Suggest Changes

What is Full Stack Development?

Full Stack

If you are already a full-stack developer you should visit the following tracks.

Frontend Backend DevOps

Target audience for this roadmap is absolute beginners wanting to get into full-stack development.

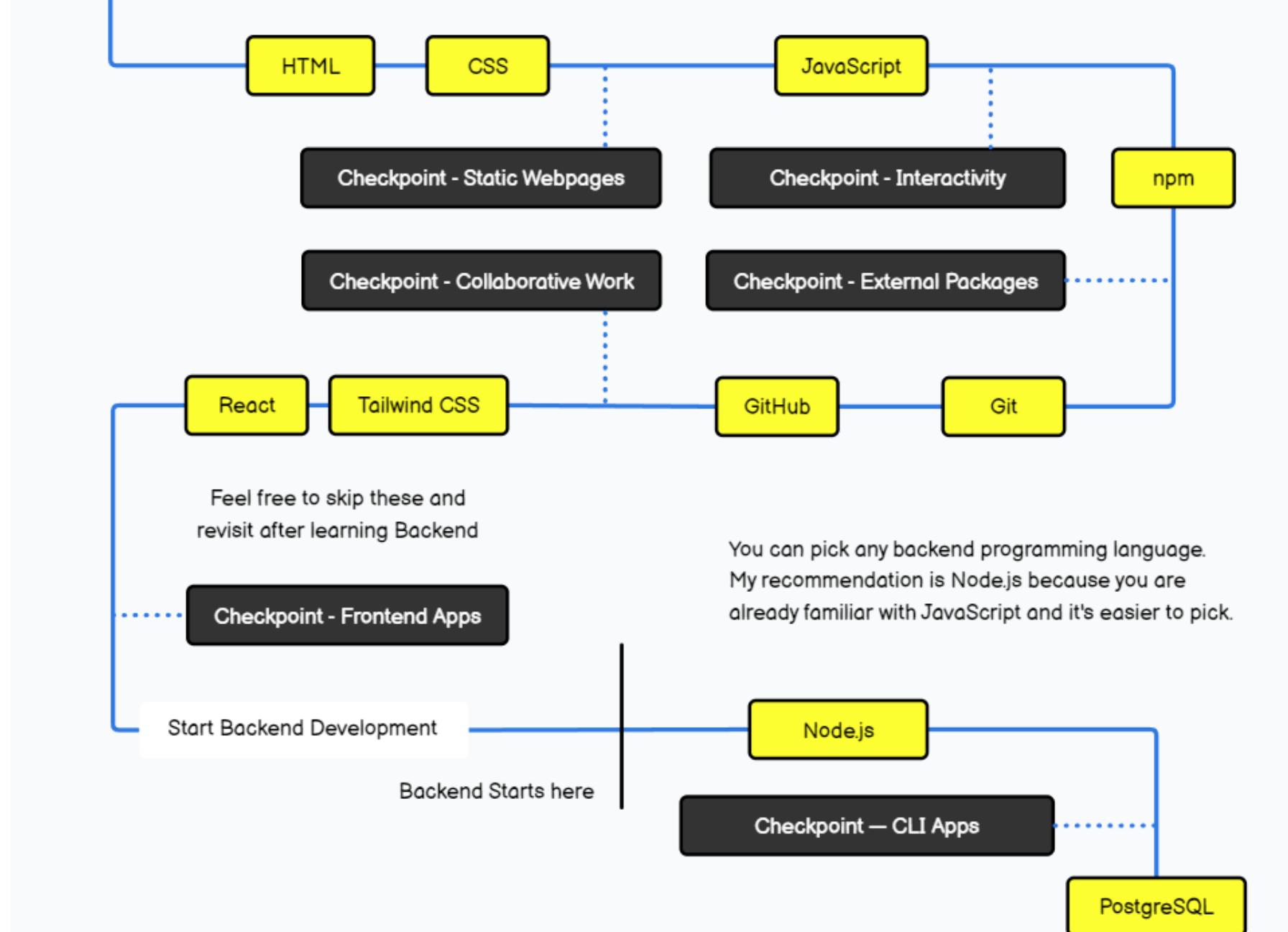
Find the detailed version of this roadmap along with other similar roadmaps

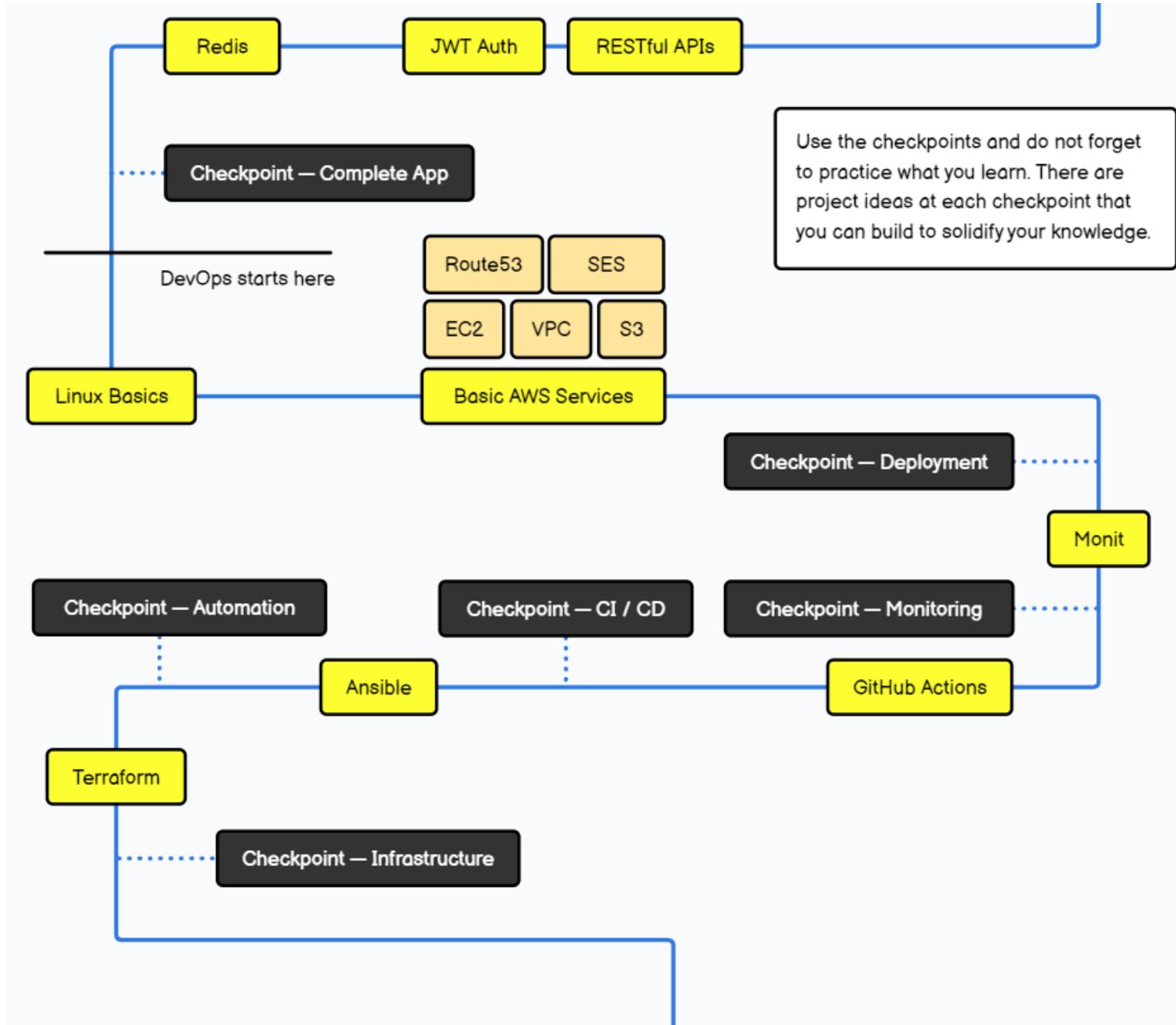
roadmap.sh

Key topics to learn

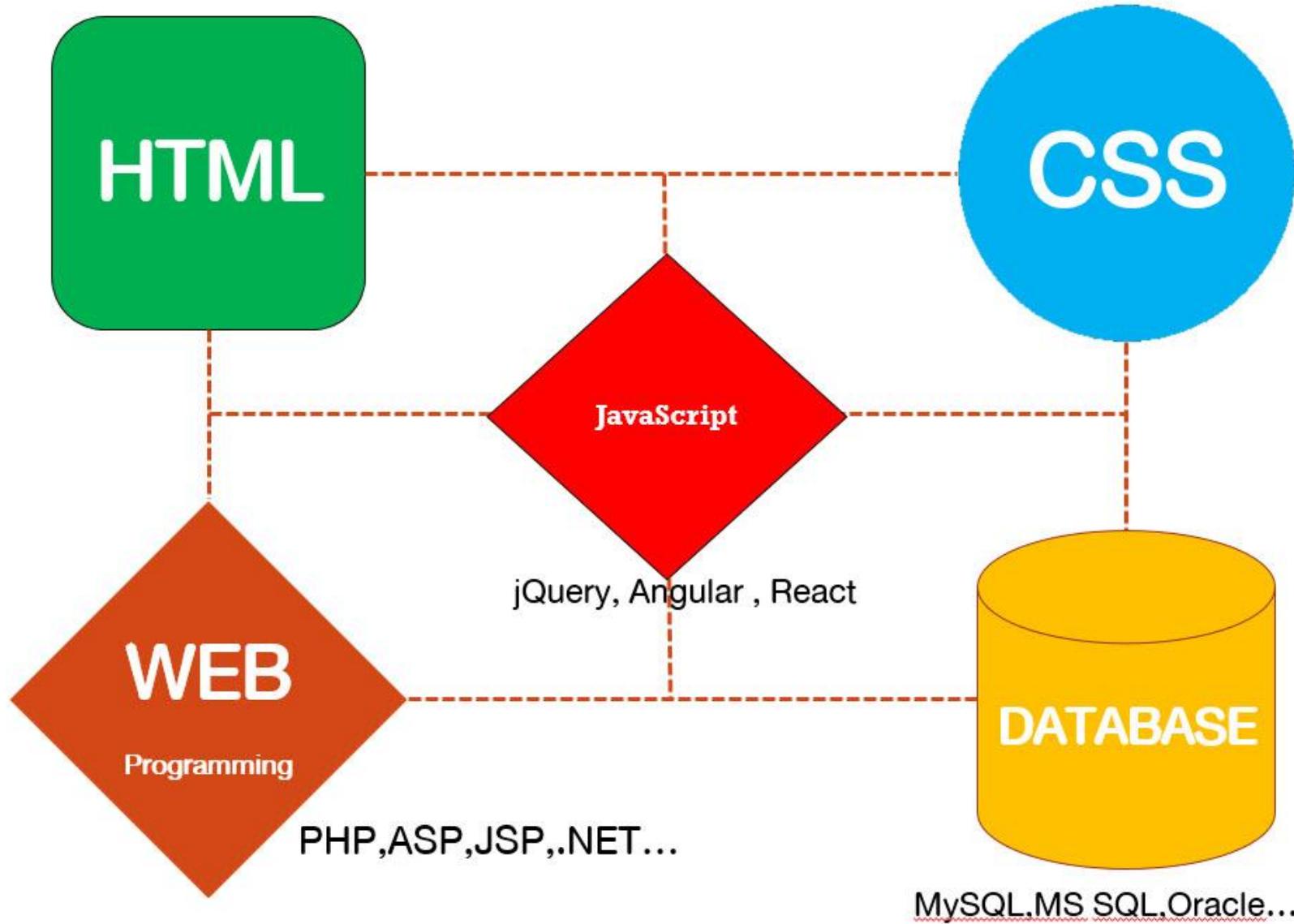
Project ideas and suggestions

ที่มา <https://roadmap.sh/full-stack>





# แนวทางการสร้างเว็บไซต์





## Section 2

### พื้นฐาน HTML 5 & CSS 3 และ JavaScript ES6

- พื้นฐาน HTML & CSS
- พื้นฐานภาษา HTML 5
- พื้นฐานภาษา CSS 3
- พื้นฐาน JavaScript
- พื้นฐาน ES6

# พื้นฐาน HTML

# What Is HTML?

HTML is a markup language for creating web documents / pages

- Hyper Text Markup Language
- Markup language – Uses a set of markup tags
- NOT a programming language

# HTML Tags

- HTML tags define HTML elements which is an individual component of an HTML document or web page.
- Keywords surrounded by angle brackets

**< p > This is a paragraph element in paragraph tags </ p >**

## Markup Language

**<head>** \_\_\_\_\_ Open tag (แท็กเปิด)

•

•

**</head>** \_\_\_\_\_ Close tag (แท็กปิด)

Paired Tags (แท็กคู่)

## Markup Language

<img>

Open tag (แท็กเปิด)

<br> <hr> <input> <link> <meta>



Single Tags (แท็กเดียว)

# HTML Boilerplate

## แม่แบบพื้นฐานของหน้าเว็บ HTML

```
<!doctype html>
<html lang="th">
<head>
    <meta charset="utf-8">
    <title>The HTML5</title>
    <meta name="descripttion" content="The HTML5 template">
    <link rel="stylesheet" href="css/styles.css">
</head>
<body>
    ... Content here ...
    <script src="js/scripts.js"></script>
</body>
</html>
```

# Page Structure

```
<!DOCTYPE html>
<html>
    <head>
        <title>My Webpage</title>
    </head>
    <body>
        <p>This is my first web page</p>
    </body>
</html>
```

# Page Structure

**<!DOCTYPE html>** - Declaration defines this document as  
HTML5

**<html>** - Describes an HTML document

**<head>** - Provides information about the document

**<title>** - Provides a title for the document

**<body>** - Describes the visible page content

# Common Doctype Declarations

## HTML5

```
<!DOCTYPE html>
```

## HTML 4.01

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"  
"http://www.w3.org/TR/html4/loose.dtd">
```

## XHTML 1.0

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"  
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

# Closing Tags

Not all tags have closing tags like `<p></p>`

Some tags can close themselves

Line-Break Tag - `<br>`

Self Closing Line-Break Tag - `<br />` (*Remnant of XHTML*)

Paragraph <p>

Headings <h1> - <h6>

Link <a>

Image <img>

Unordered List <ul>

Ordered List <ol>

List Item <li>

Table <table>

Form <form>

Input <input>

Button <button>

Line Break <br>

Horizontal Rule <hr>

Division <div>

HTML <html>

Head <head>

Body <body>

Strong <strong>

# Attributes

Tags can also have “attributes” which include extra bits of information about that tag/element. Attributes appear in the tag inside of quotation marks

**<tag attribute=“value”></tag>**

**<a href=“http://google.com”>Click</a>**

**<h1 id=“myHeading”></h1>**

# **Block Level Elements**

- Always starts on a new line
- Takes up full available width

**<div>**

**<h1> - <h6>**

**<p>**

**<form>**

# Inline Level Elements

- Does not start on a new line
- Takes only width necessary

**<span>**

**<a>**

**<img>**

# คำสั่ง HTML พื้นฐาน ที่ควรทำความรู้จัก

#	Markup	Type	Description
1.	<!Doctype>	Single	
2.	<html>...</html>	Paired	
3.	<head>...</head>	Paired	
4.	<body>...</body>	Paired	
5.	<title>...</title>	Paired	
6.	<meta>	Single	
7.	<link>	Single	
8.	<script>...</script>	Paired	

#	Markup	Type	Description
9	<!-- some text -->	Single	
10.	<div>...</div>	Paired	
11.	<span>...</span>	Paired	
12.	<p>...</p>	Paired	
13.	 	Single	
14.	<hr>	Single	
15.	<h1>...</h1> - <h6>..</h6>	Paired	
16.	<img>	Single	
17.	<ul> <li>...</li> </ul>	Paired	

#	Markup	Type	Description
17.	<ul> <li>...</li> </ul>	Paired	
18.	<ol> <li>...</li> <ol>	Paired	
19.	<table> ... <table>	Paired	
20.	<form>...</form>	Paired	
21.	<input>	Single	
22.	<header>...</header>	Paired	
23.	<footer>...</footer>	Paired	

# คำสั่งพื้นฐานของ HTML

<!--ข้อความ-->

Comment หรือการอธิบาย

<br>

Breakline การตัดขึ้นบรรทัดใหม่

<p> ข้อความ </p>

Paragraph การจัดย่อหน้าใหม่

<hr width = "50%" size = "3">

Horizontal line การวาดเส้นในแนวนอน

&ampnbsp

None break space การเว้นช่องไฟ

<img src = “images/photo.jpg”>

Tag คำสั่งแสดงผลรูปภาพ

<center> ข้อความ </center>

คุณสมบัติการจัดวัตถุกึ่งกลางจอ

# การจัดการห้องความ

# จัดรูปแบบข้อความที่เป็นหัวเรื่อง

<h1>....</h1>

<h2>....</h2>

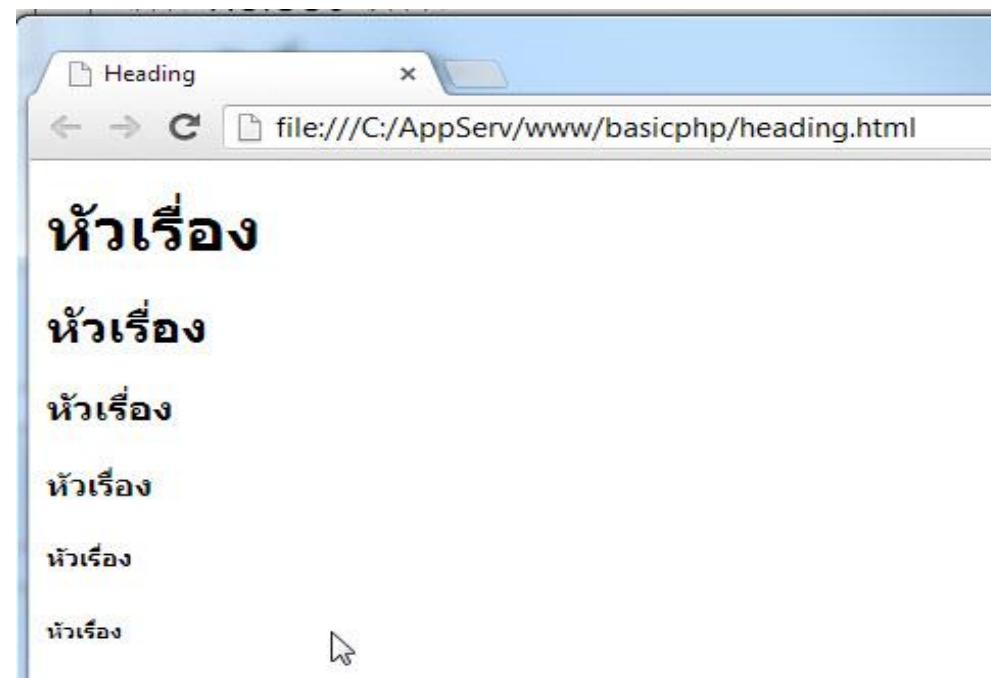
<h3>....</h3>

<h4>....</h4>

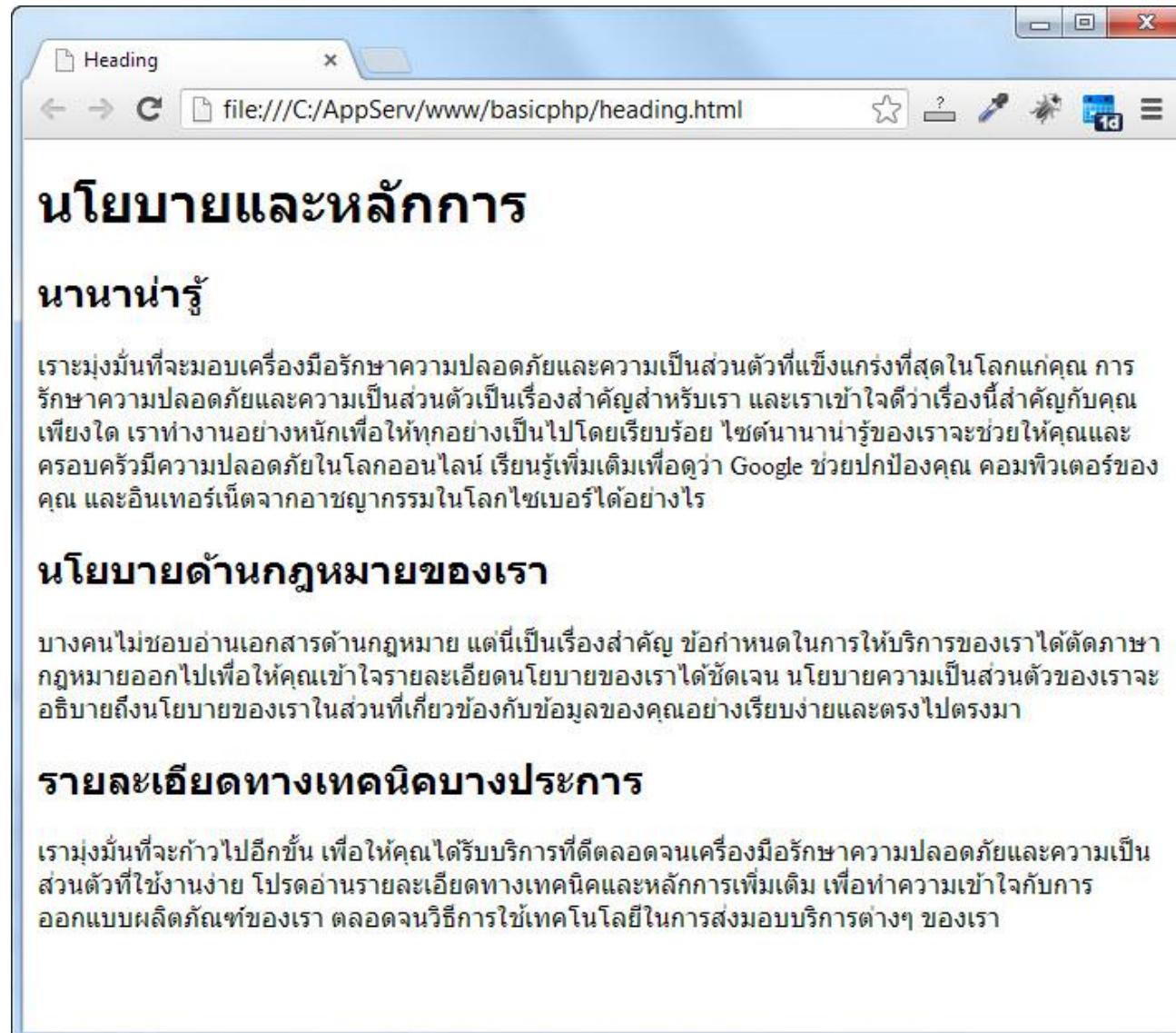
<h5>....</h5>

<h6>....</h6>

<h1>หัวเรื่อง</h1>  
<h2>หัวเรื่อง</h2>  
<h3>หัวเรื่อง</h3>  
<h4>หัวเรื่อง</h4>  
<h5>หัวเรื่อง</h5>  
<h6>หัวเรื่อง</h6>



# ตัวอย่างการใช้ <h1> ร่วมกับแท็ก <p>



# จัดรูปแบบตัวอักษร (ขนาด, สี, หนา, เอียง)

<font size = "3"> ข้อความ </font>

<font color = "red"> ข้อความ </font>

<font face = "Arial"> ข้อความ </font>

<b> ข้อความ </b>

<i> ข้อความ </i>

<u> ข้อความ </u>

# ຈຸດເສື່ອມໂຍງຂ້ອນຸລ

```
<a href ="#news"> Hot News </a>, <a name ="news">
```

```
<a href ="news.html"> Hot News </a>
```

```
<a href ="http://www.thai.com"> Thai </a>
```

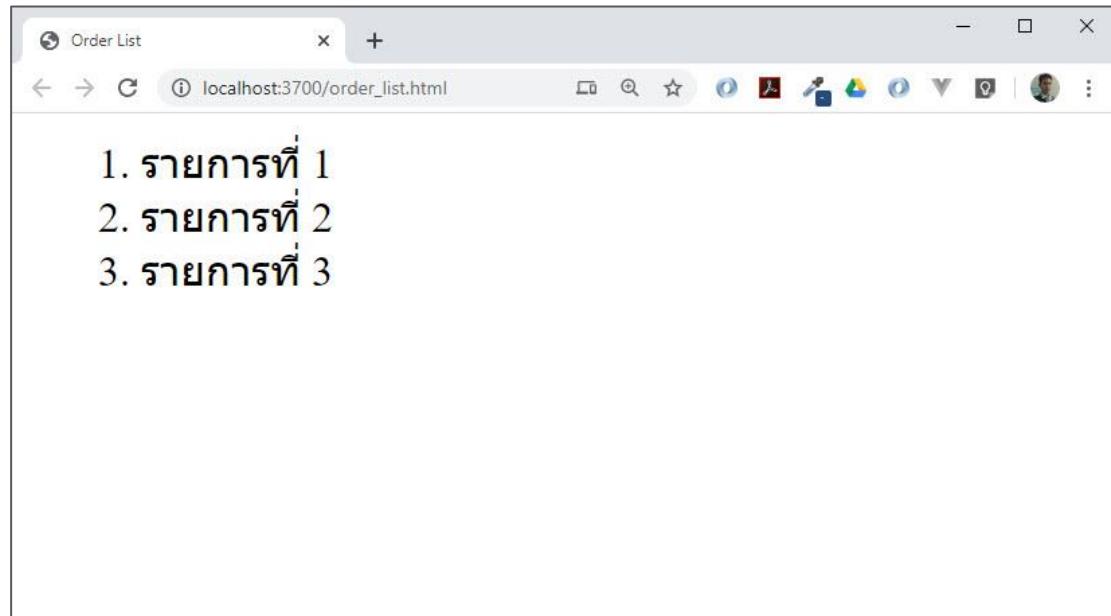
```
<a href ="http://www.thai.com" target = "_blank" > Thai </a>
```

```
<a href ="mailto:yo@mail.com"> Email </a>
```

```
<a href ="tel:08989389389"> Call Us </a>
```

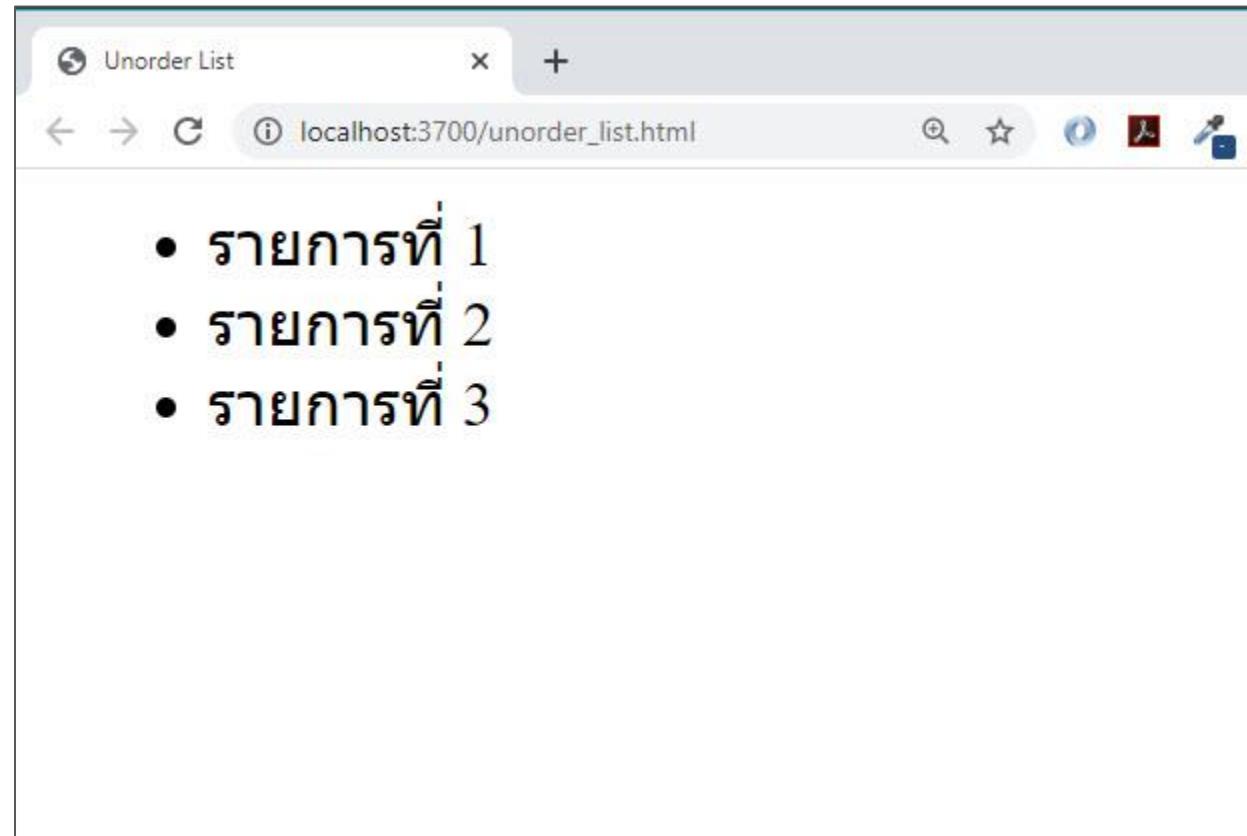
# การแสดงผลรายการแบบมีหมายเลขกำกับ

```
<body>
<ol>
    <li>รายการที่ 1</li>
    <li>รายการที่ 2</li>
    <li>รายการที่ 3</li>
</ol>
</body>
```



# การแสดงผลรายการแบบสัญลักษณ์

```
<body>
<ul>
    <li>รายการที่ 1</li>
    <li>รายการที่ 2</li>
    <li>รายการที่ 3</li>
</ul>
</body>
```



# การสร้างตารางใน HTML

table head data <th>

table <table>

table body  
<tbody>

ID	Name	Email
1	John	john@email.com
2	Jack	jack@email.com
3	Joe	joe@email.com
4	Jane	jane@email.com

table head  
<thead>

table row <tr>

table data <td>

# ตัวอย่างคำสั่งสร้างตารางใน HTML

ID	Name	Email
1	John	john@email.com
2	Jack	jack@email.com
3	Joe	joe@email.com
4	Jane	jane@email.com

```
<table>
  <thead>
    <tr>
      <th>ID</th>
      <th>Name</th>
      <th>Email</th>
    </tr>
  </thead>
  <tbody>
    ...
    <tr>
      <td>4</td>
      <td>Jahn</td>
      <td>john@email.com</td>
    </tr>
    ...
  </tbody>
</table>
```

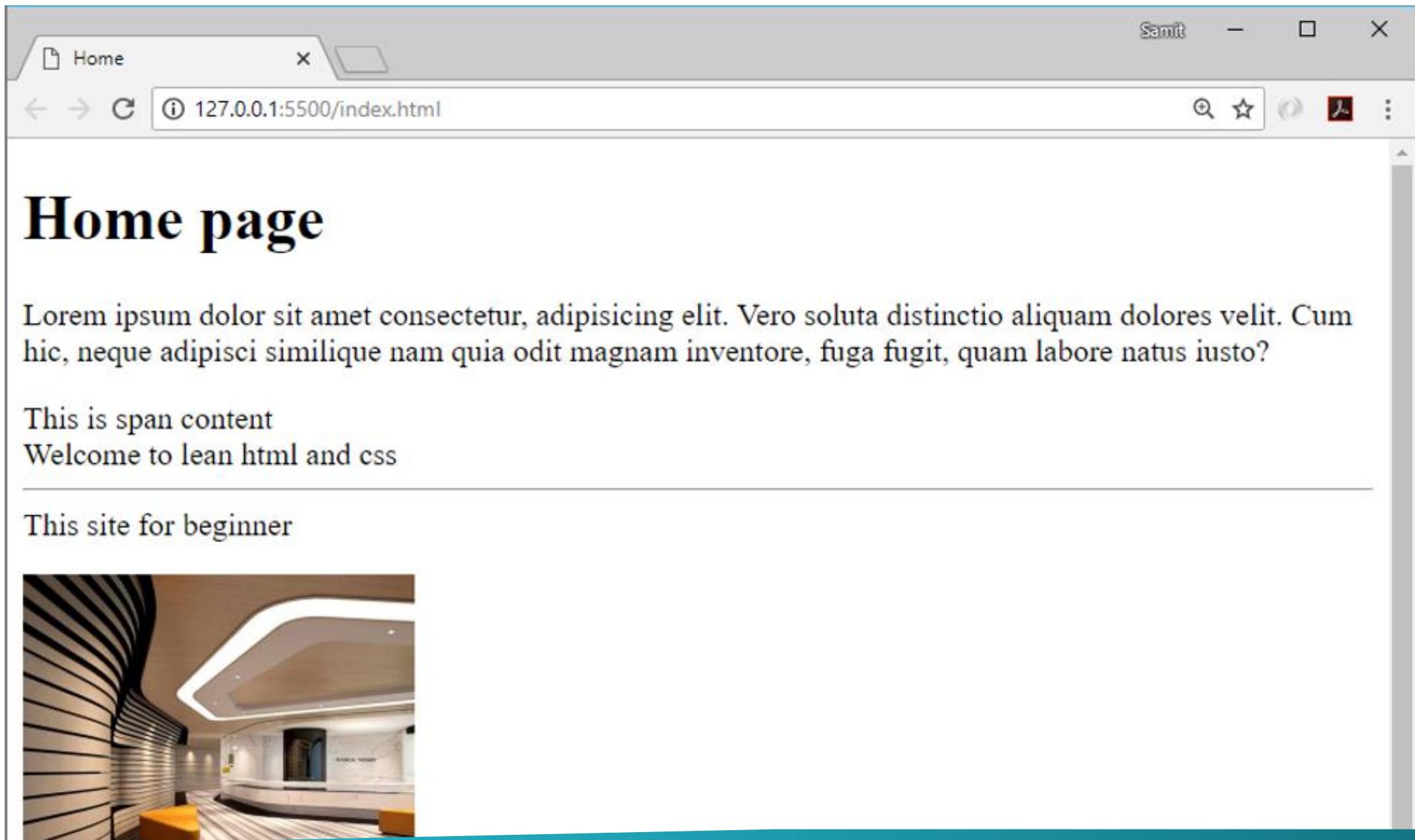
# การทำงานกับฟอร์มต่าง ๆ

Name	Value
Name	<input type="text"/>
Sex	<input type="radio"/> Male <input checked="" type="radio"/> Female
Eye color	green <input type="button" value="▼"/>
Check all that apply	<input type="checkbox"/> Over 6 feet tall <input type="checkbox"/> Over 200 pounds
Describe your athletic ability:	
<input type="button" value="Enter my information"/>	

# ทดลองสร้างหน้าเว็บเพจด้วย html tag

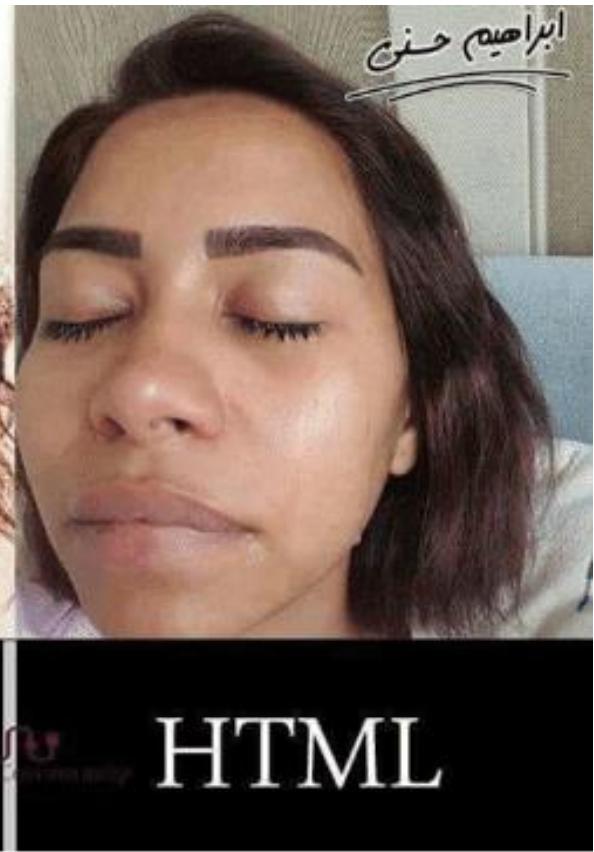
## index.html

```
<body>
  <!-- ตัวอย่างข้อความคำอธิบาย-->
  <h1>Home page</h1>
  <p>Lorem ipsum dolor sit amet consectetur, adipisicing elit. Vero soluta distinctio aliquam dolores velit. Cum hic, neque adipisci similique nam quia odit magnam inventore, fuga fugit, quam labore natus iusto?</p>
  <div>
    <span>This is span content</span>
    <br>
    Welcome to lean html and css
    <hr>
```



# พื้นฐาน CSS

# รู้จักภาษา CSS (Cascading Style Sheet)



Jordy Gabriel

#Jdam

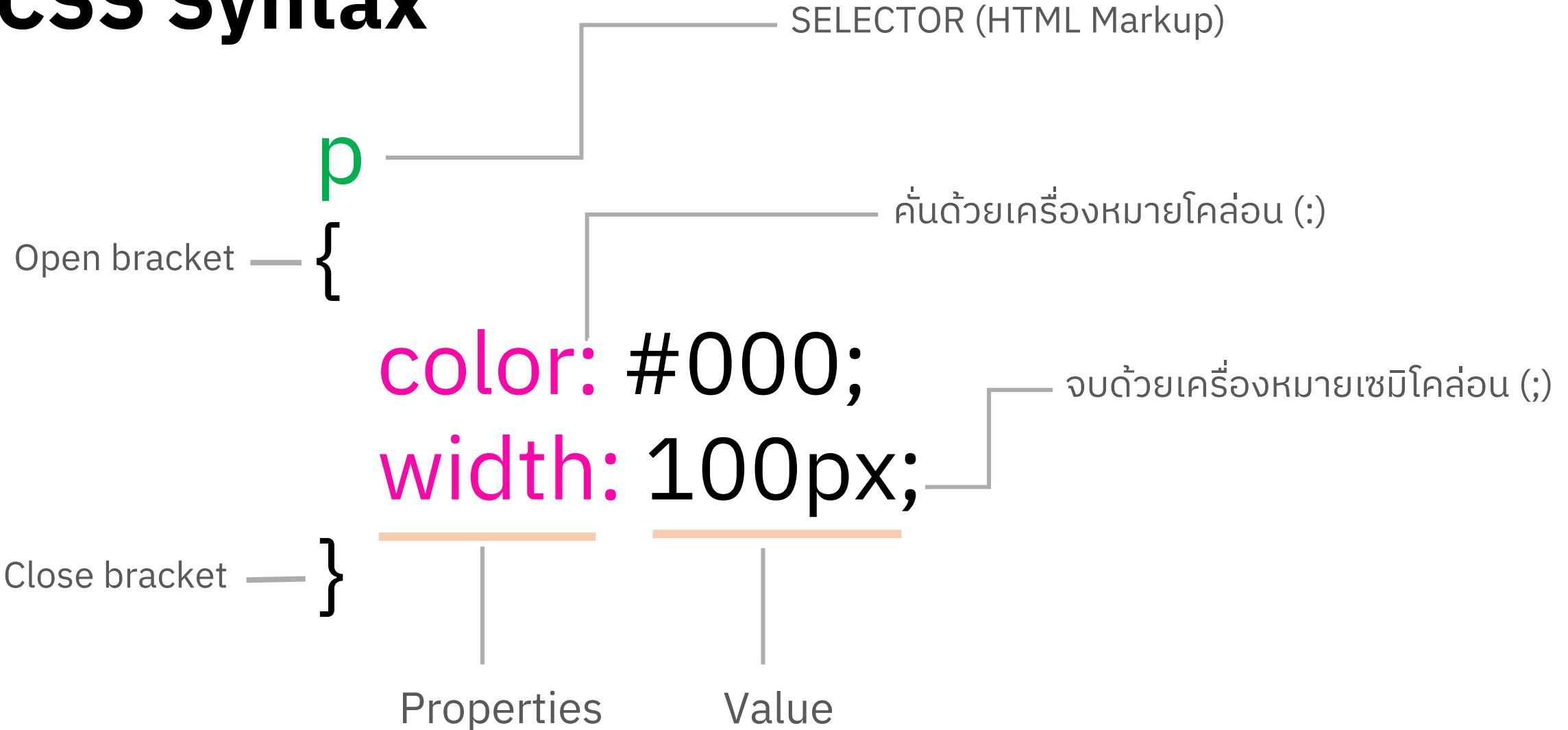
# CSS

## Cascading Style Sheet

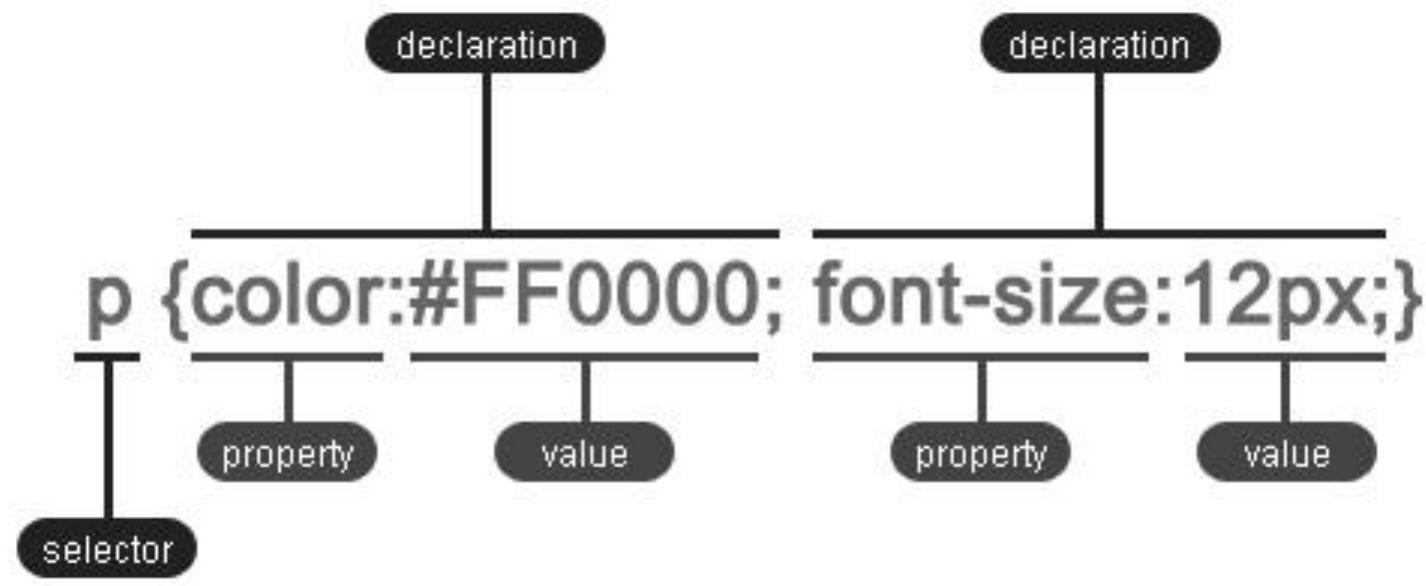
มีความสำคัญอย่างมากต่อการตกแต่ง ความเรียบร้อย สวยงามของเว็บไซต์ โดยตัวภาษาเป็นลักษณะการทำงานร่วมกับ HTML tag หรือการจัดการกับ markup ของ html นั่นเอง ภาษา CSS จะต้องทำงานร่วมกับ HTML ไม่สามารถทำงานลำพังได้

# โครงสร้างและรูปแบบ ของภาษา CSS

# CSS Syntax



# ส่วนประกอบของ CSS



ภาพจาก [www.css-learning.com](http://www.css-learning.com)

- Selector คือ แก๊คหรือสิ่งที่ต้องการให้ css แสดงผลกับมัน
- Property คือ รูปแบบของ CSS ที่ต้องการแสดงผล
- Value คือ ค่าที่เราต้องการกำหนด ว่าเพิ่มเติมได้ที่  
<http://www.w3schools.com/css/default.asp>

# ID Selector

HTML

```
<div id="nav">  
.  
.  
</div>
```

CSS

```
#nav  
{  
    width: 100px;  
}
```

# Class Selector

HTML

```
<div class="nav">  
.  
.  
</div>
```

CSS

```
.nav  
{  
    width: 100px;  
}
```

# ຕຳແໜ່ງ CSS ແລະ ຮະດັບຄວາມສໍາຄັญ

### Inline markup css

```
<div style="color:#000">
```

1

### Internal markup css

```
<style type="text/css">  
div  
{  
    color:#000;  
}  
</style>
```

2

### External markup css

```
<link rel="stylesheet" href="css/style.css"
```

3

# การเขียน CSS แบ่งออกเป็น 3 รูปแบบ

## 1. เขียนในบรรทัด (inline)

เป็นการเขียนเพื่อกำหนดรูปแบบของแท็กเพียงเล็กน้อยเท่านั้น ไม่ควรใช้งานกับเว็บที่มีโครงสร้างใหญ่ เพราะจะทำให้ลำบากในการแก้ไข

```
<div style= "background-color: red;">.....</div>
```

## 2. เขียนแบบหัวเว็บ (internal)

การเขียนประเภทนี้จะมาสำหรับการกำหนดรูปแบบเว็บไซต์หน้าได้  
หน้าหนึ่ง โดยจะถูกเขียนไว้ในส่วนของแท็ก <head>...</head>

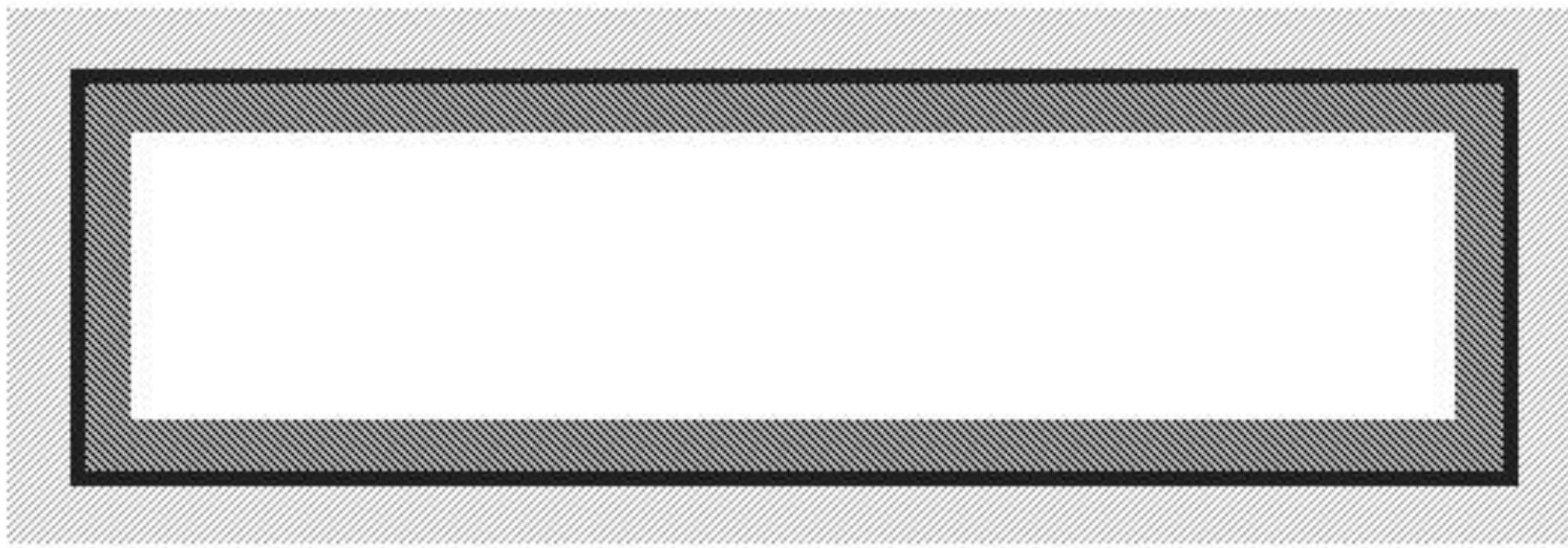
```
<head>
    <style type= "text/css" >
        .....คำสั่ง CSS .....
    </style>
</head>
```

### 3. การเขียนໄວ้ภายนอก (External)

เป็นที่นิยมใช้ในปัจจุบัน มากที่สุด เนื่องจากสามารถรับเว็บที่มีโครงสร้างใหญ่ การเรียกใช้งานจะถูกเรียกผ่านแท็ค <link> ดังตัวอย่าง

```
<head>
<link rel= "stylesheet" type="text/css" href= "style.css" />
</head>
```

# CSS Box Model



margin



border



padding

# Padding & Margin

**margin:25px;**

all four margins are 25px

**margin:25px 50px;**

top and bottom margins are 25px  
right and left margins are 50px

**margin:25px 50px 75px;**

top margin is 25px  
right and left margins are 50px  
bottom margin is 75px

**margin:25px 50px 75px 100px;**

top margin is 25px  
right margin is 50px  
bottom margin is 75px  
left margin is 100px

**padding:25px;**

all four paddings are 25px

**padding:25px 50px;**

top and bottom paddings are 25px  
right and left paddings are 50px

**padding:25px 50px 75px;**

top padding is 25px  
right and left paddings are 50px  
bottom padding is 75px

**padding:25px 50px 75px 100px;**

top padding is 25px  
right padding is 50px  
bottom padding is 75px  
left padding is 100px

# คำสั่งที่ถูกเรียกใช้งานบ่อยครั้ง

- float : กำหนดให้วัตถุลอย
- margin : กำหนดพื้นที่ที่อยู่นอกเส้นขอบ
- padding : กำหนดพื้นที่ที่อยู่ในเส้นขอบ
- background : กำหนดคุณสมบัติพื้นหลัง
- border : กำหนดคุณสมบัติเส้นขอบ
- font : กำหนดคุณสมบัติตัวหนังสือ
- width : กำหนดความกว้าง
- height : กำหนดความสูง

# WORKSHOP

main

LOGO

logo

header

menu

Home About Service Contact

slide

ສູກາວ

content

subcontent

subcontent

subcontent

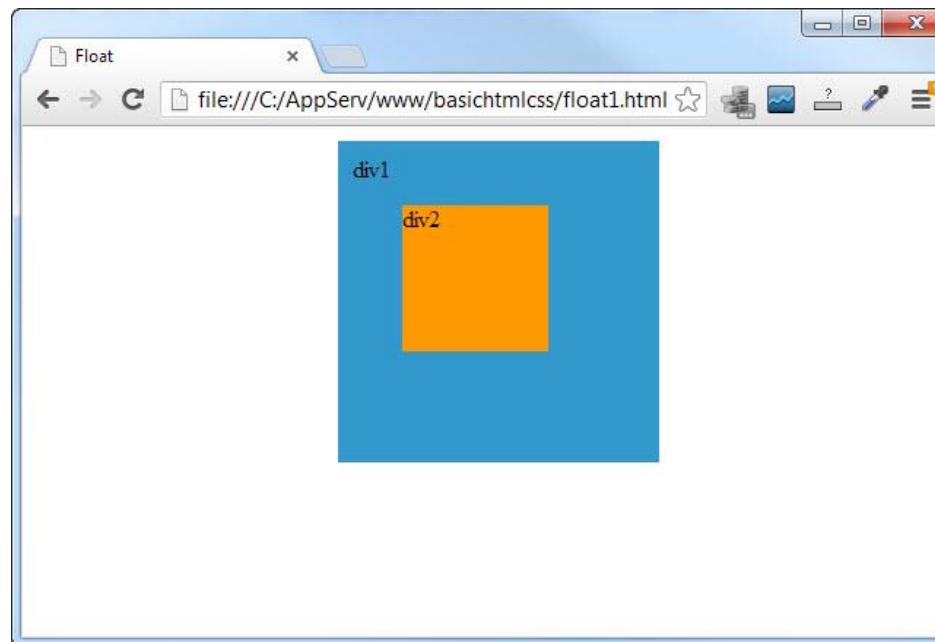
# Floating & Positioning

# ข้อควรรู้

ตามปกติการจัดเรียง Element ต่าง ๆ ในหน้าเว็บจะวางแบบ Normal Flow และจะต้องอยู่ภายใน Container เสมอ โดยเรียงจากบนลงล่าง และจากซ้ายไปขวา

แต่ในบางครั้งเราไม่ต้องการวางแบบ Normal Flow เราสามารถกำหนดได้เองโดยใช้คำสั่ง Float และ Position

```
10 }
11 #div1{
12   padding:10px;
13   background: #39C;
14   width:200px;
15   position:relative;
16   left:50%; /*วางตั้งกลางหน้า*/
17   top:10px;
18   margin-left: -100px; /*เลื่อนไปด้านซ้ายอีก 100px*/
19   height: 200px;
20 }
21 #div2 {
22   background: #F90;
23   width:100px;
24   height:100px;
25   position: absolute; /*อยู่ภายใน container*/
26   top: 20%; /*วางต่ำลงมาจากด้านบน 20%*/
27   left:20%; /*วางต่ำลงจากด้านซ้าย 20%*/
28 }
```

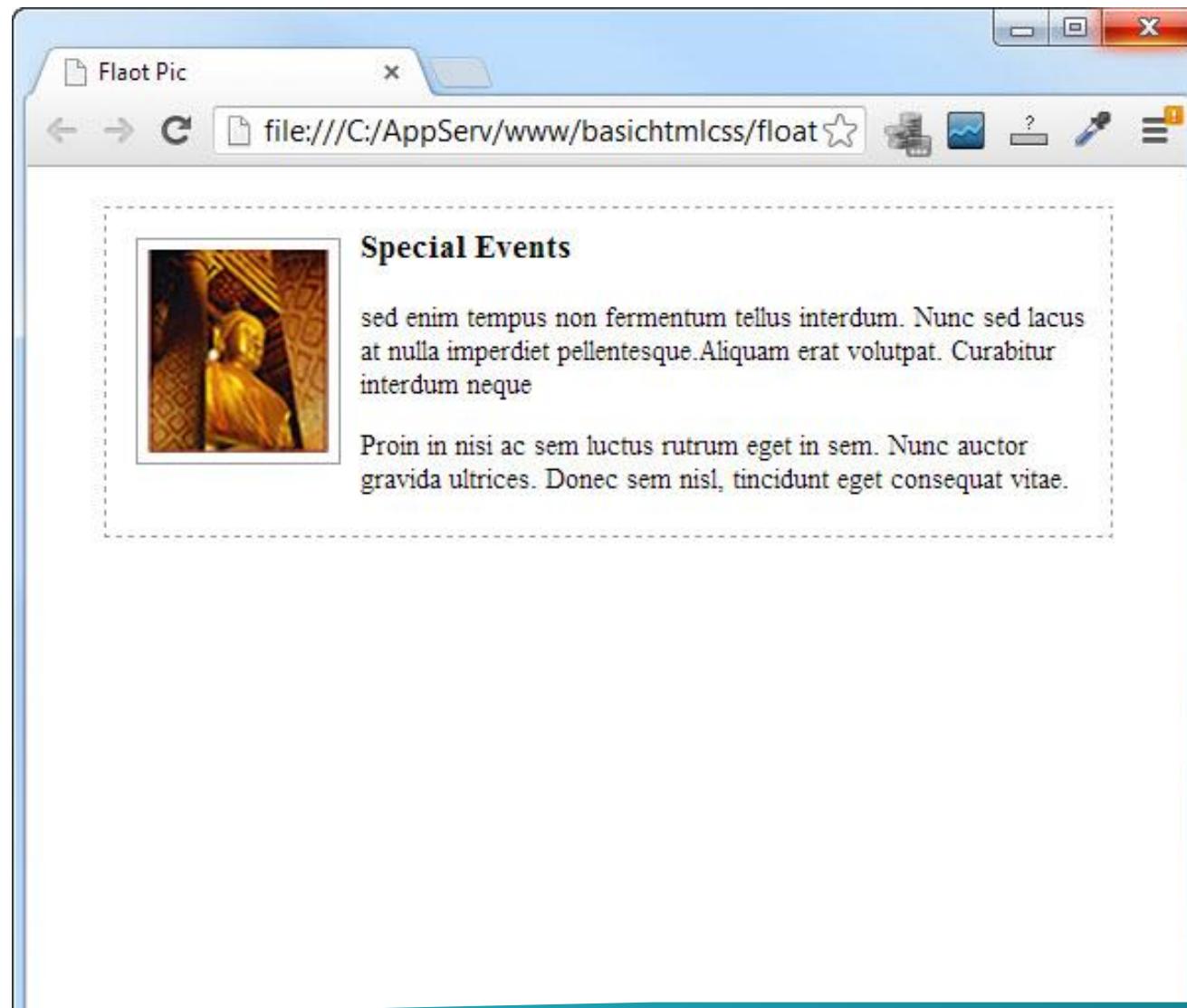


# Floating

การ Float Element ทำให้ Element นั้นออกจาก Normal Flow คือใช้ตำแหน่ง Default ของตัวเองเป็นหลัก และผลักออกไปชิดซ้ายหรือขวาของ Container ถ้าไม่มีที่เหลือพожะขึ้นบรรทัดใหม่

จำไว้ว่าการ Float จะมีผลกับ Element กี่ตัวมาเสมอซึ่งจะ Float ตามตัวก่อนหน้าในแบบเดียวกัน เช่นชิดซ้ายหรือขวาไปเรื่อยๆ จนกว่ามีคำสั่งให้หยุด การ float ด้วยการ Clear float

# ตัวอย่างการ Float ภาพพร้อมข้อความ



# Positioning

เป็นการจัดตำแหน่งของ Element ใด ๆ ในบริเวณที่อย่างเป็นอิสระจาก Normal Flow โดยจะต้องบอกว่าใช้ Container อะไรเป็นหลัก เช่น ใช้ตำแหน่งข้างบนสุดของบริเวณที่เป็นหลัก หรือ Element ที่วางก่อนหน้าเป็นหลัก หรือ Container ของตัวเอง (ตัวใกล้สุดที่ครอบอยู่) เป็นหลัก

## Container

ตามปกติการวาง Element จะต้องอยู่ใน Container เสมอ ซึ่งจะเป็นไปได้ดังนี้ใช้หน้าบริเวณที่เป็น Container เวลาจะ position:fixed ใช้ Container อื่นๆ กับ position relative ,static หรือ absolute

# Position Property มี 4 แบบ

## 1. static

เป็นค่า Default คือว่างแบบ Normal Flow ซึ่งเราไม่สามารถปรับตำแหน่งด้วยค่า offset ใด ๆ ได้เลย แต่สามารถกำหนดค่า margin ได้ตามปกติ

## 2. fixed

เป็นการวางที่ตายตัว และออกจาก Normal Flow เลยใช้บราวน์เซอร์เป็นหลัก โดยกำหนด ตำแหน่งที่ต้องการจากมุมซ้ายสุด การกำหนดแบบนี้จะอยู่คงที่ไม่ว่าจะเลื่อนスクอร์บาร์ก็ตาม

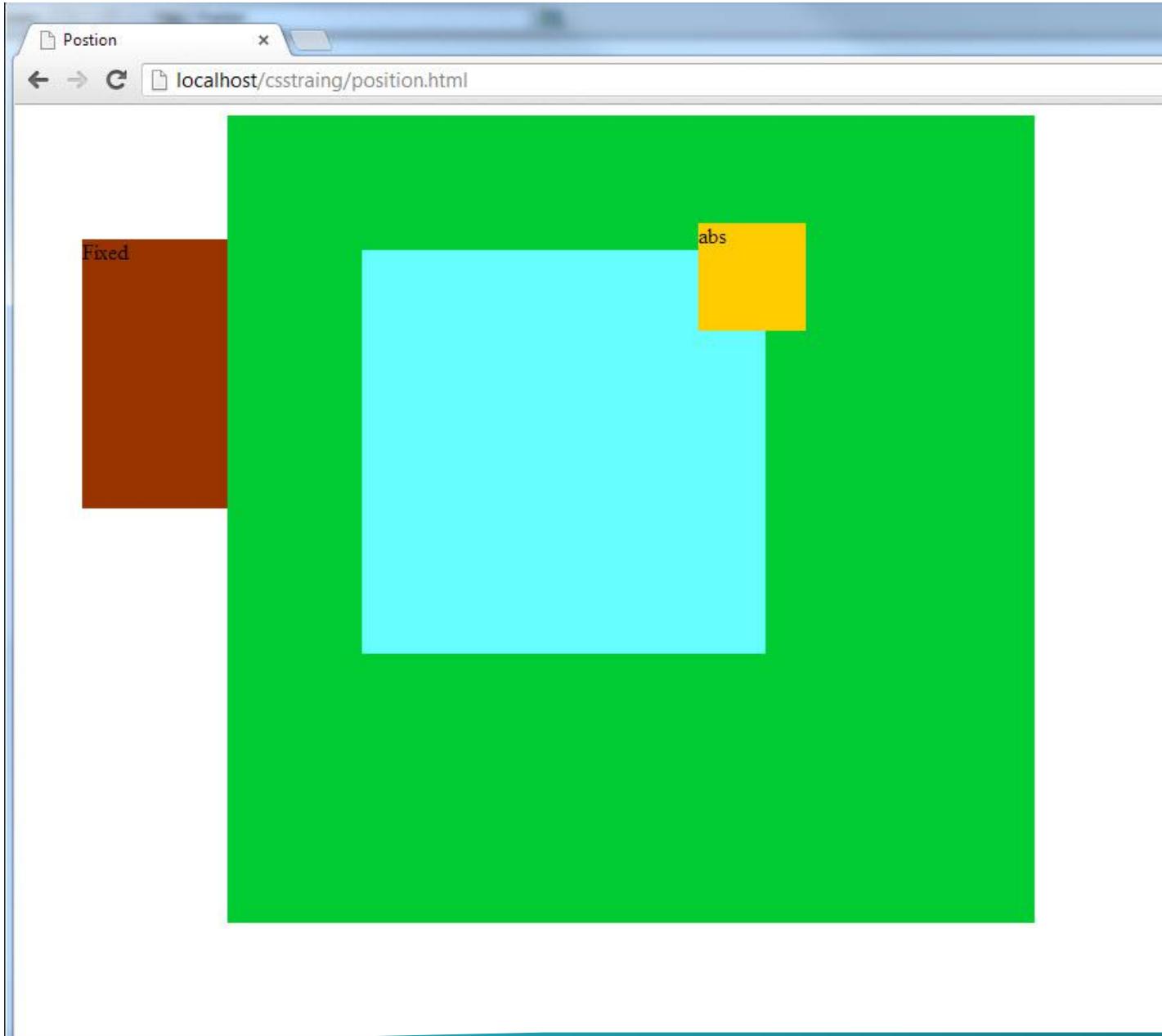
## **3.relative**

วางแผนคล้ายกับ Static กี่ใช้ตำแหน่ง Default ของตัวเองจากระบบ Normal Flow เป็นหลักแต่สามารถปรับตำแหน่งด้วย offset ได้

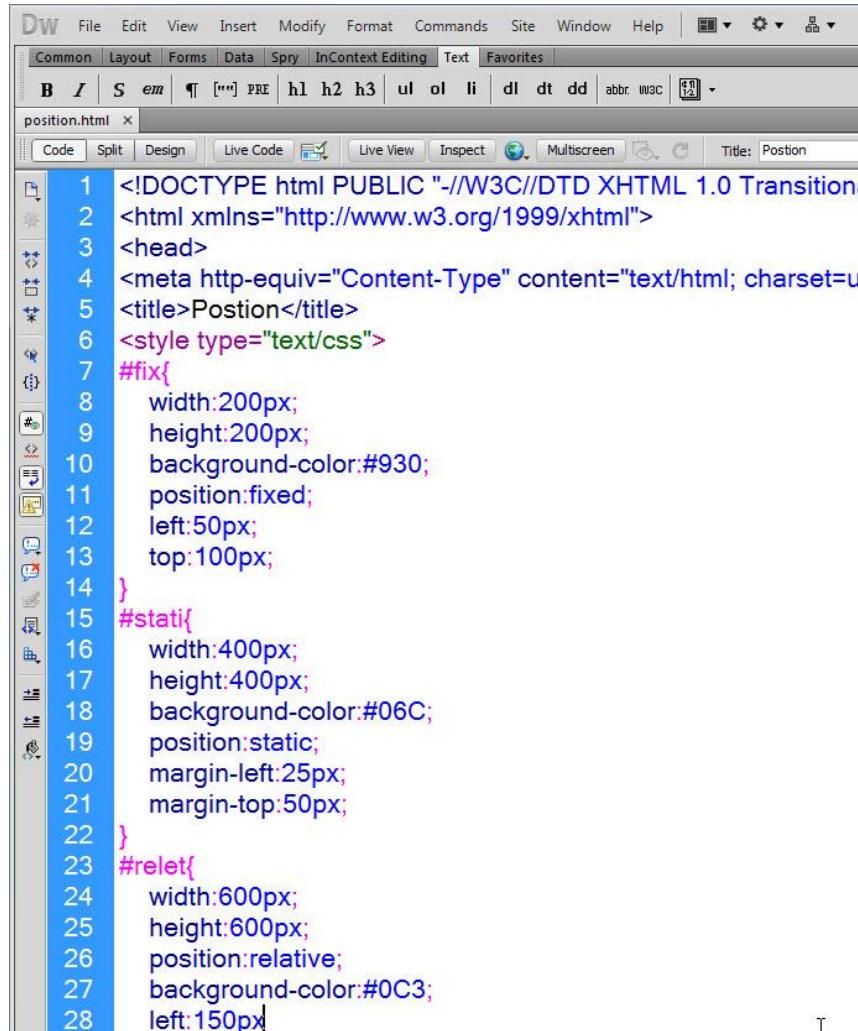
## **4.absolute**

เป็นการวางแผนจาก Normal Flow ปกติโดยการใช้ตำแหน่งของ Parent Element หรือ Container ที่ใกล้ที่สุดที่ครอบ Element นั้นอยู่ เป็นหลัก และมีข้อกำหนดเพิ่มเติมว่า Parent Element ต้องมีการกำหนด position แบบอื่น ๆ ก็ไม่ใช่ static ไม่ เช่นนั้นจะอ้างอิงไปยังตัวถัดไปจนกว่าจะพบ และถ้าไม่พบก็จะใช้ browser เป็น Container แทน

# ตัวอย่าง



# Position CSS

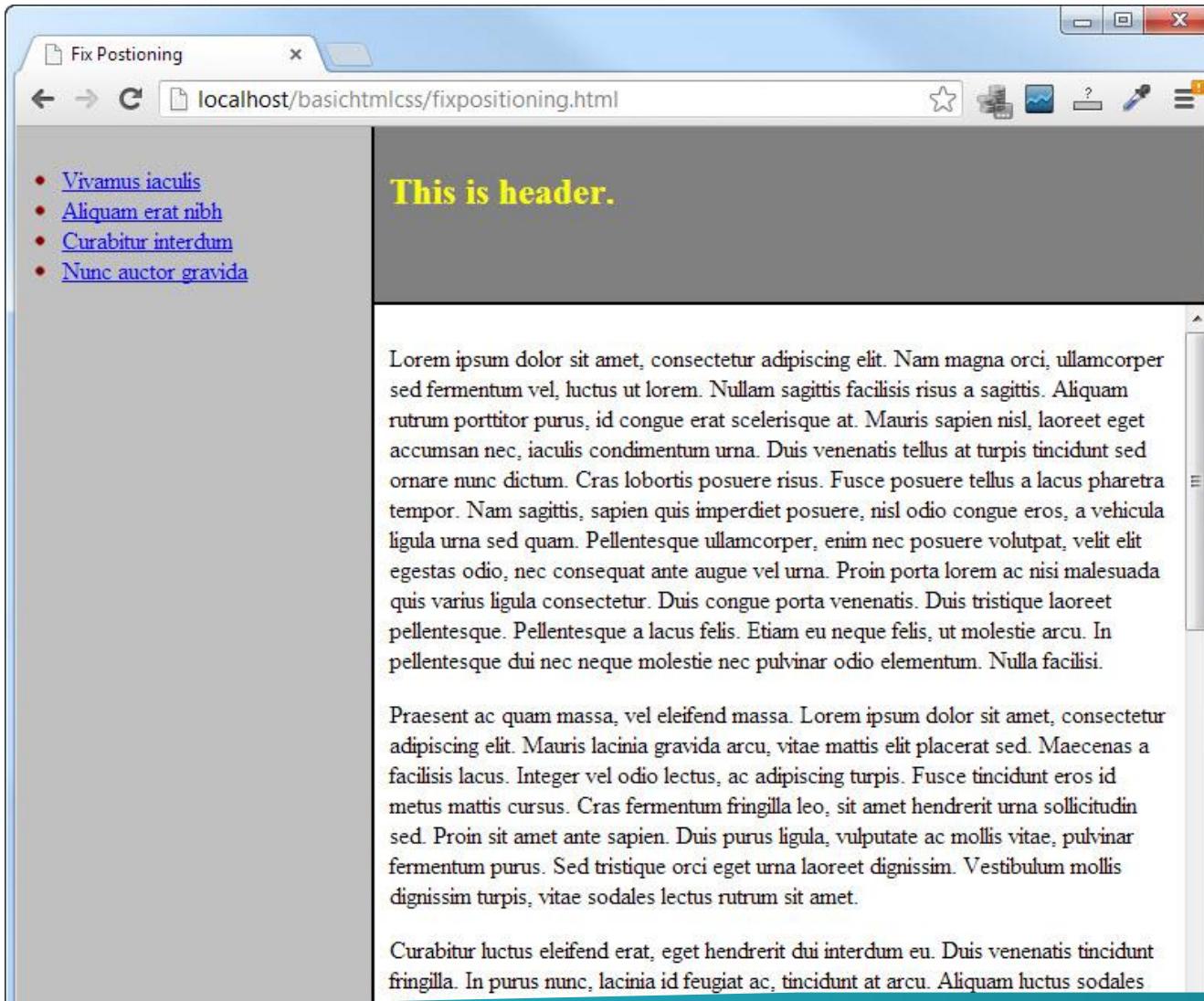


```
1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional"
2 <html xmlns="http://www.w3.org/1999/xhtml">
3 <head>
4 <meta http-equiv="Content-Type" content="text/html; charset=ut
5 <title>Postion</title>
6 <style type="text/css">
7 #fix{
8   width:200px;
9   height:200px;
10  background-color:#930;
11  position:fixed;
12  left:50px;
13  top:100px;
14 }
15 #stati{
16   width:400px;
17   height:400px;
18   background-color:#06C;
19   position:static;
20   margin-left:25px;
21   margin-top:50px;
22 }
23 #relet{
24   width:600px;
25   height:600px;
26   position:relative;
27   background-color:#0C3;
28   left:150px;
```



```
37 }
38
39 #abs{
40   width:80px;
41   height:80px;
42   position:absolute;
43   background-color:#FC0;
44   top:-20px;
45   left:250px;
46 }
47
48 </style>
49 </head>
50
51 <body>
52 <div id="fix">Fixed</div>
53 <!--<div id="stati">Static</div>-->
54 <div id="relet">
55
56 <div id="relet2">
57 <div id="abs">abs</div>
58 </div>
59
60 </div>
61
62 </body>
63 </html>
```

# ตัวอย่าง Position Fixed



```
DW File Edit View Insert Modify Format Commands Site Window Help  
Common Layout Forms Data Spry InContext Editing Text Favorites  
B I S em ¶ [""] PRE h1 h2 h3 ul ol li dl dt dd abbr w3c  
position.html x fixpositioning.html x  
Code Split Design Live Code Live View Inspect Multiscreen  
1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"  
2 <html xmlns="http://www.w3.org/1999/xhtml">  
3 <head>  
4 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />  
5 <title>Fix Postioning</title>  
6 <style type="text/css">  
7   body {  
8     background: black;  
9   }  
10  div#header {  
11    position: fixed;  
12    top: 0;  
13    bottom: 80%;  
14    left: 30%;  
15    right: 0;  
16    background: gray;  
17    margin-bottom: 2px;  
18    color: yellow;  
19    padding: 10px  
20  }  
21  div#sidebar {  
22    position: fixed;  
23    top: 0;  
24    bottom: 0;  
25    left: 0;  
26    right: 70%;  
27    background: silver;  
28    margin-right: 2px;  
29    color: maroon;
```

```
DW File Edit View Insert Modify Format Commands Site Window Help  
Common Layout Forms Data Spry InContext Editing Text Favorites  
B I S em ¶ [""] PRE h1 h2 h3 ul ol li dl dt dd abbr w3c  
position.html x fixpositioning.html x  
Code Split Design Live Code Live View Inspect Multiscreen Title: Fix Posti  
34   top: 20%; bottom: 0;  
35   left: 30%; right: 0;  
36   overflow: auto;  
37   background: white;  
38   color: black;  
39   padding: 10px  
40 }  
41 ul{  
42   margin-left: -20px  
43 }  
44 </style>  
45 </head>  
46 </body>  
47 <div id="header">  
48   <h2>This is header.</h2>  
49 </div>  
50 <div id="sidebar">  
51   <ul>  
52     <li><a href="#">Vivamus iaculis</a></li>  
53     <li><a href="#">Aliquam erat nibh</a></li>  
54     <li><a href="#">Curabitur interdum</a></li>  
55     <li><a href="#">Nunc auctor gravida</a></li>  
56   </ul>  
57 </div>  
58 <div id="main">  
59   <p>Lorem ipsum dolor sit amet, consectetur adipisc  
60   risus a sagittis. Aliquam rutrum porttitor purus, id congue er
```

# Z-index

เป็นการบอกลำดับของ Element ว่าจะเรียงอย่างไรในแนวแกน z เนื่องจากการจัดแบบ Absolute อาจทำให้ Element วางซ้อนกันได้ ดังนั้นเราสามารถเรียงลำดับ Element ที่ซ้อนกันนั้นเป็นชั้น ๆ ว่าอะไรมาต่อหนัง ด้วยการกำหนดค่า z-index เป็นตัวเลข เป็น 1 ได้ถ้าค่าบวกและค่าลบ โดยตัวเลขที่มีค่าน้อยกว่าจะอยู่ล่างตัวที่มีค่า z-index มากกว่า

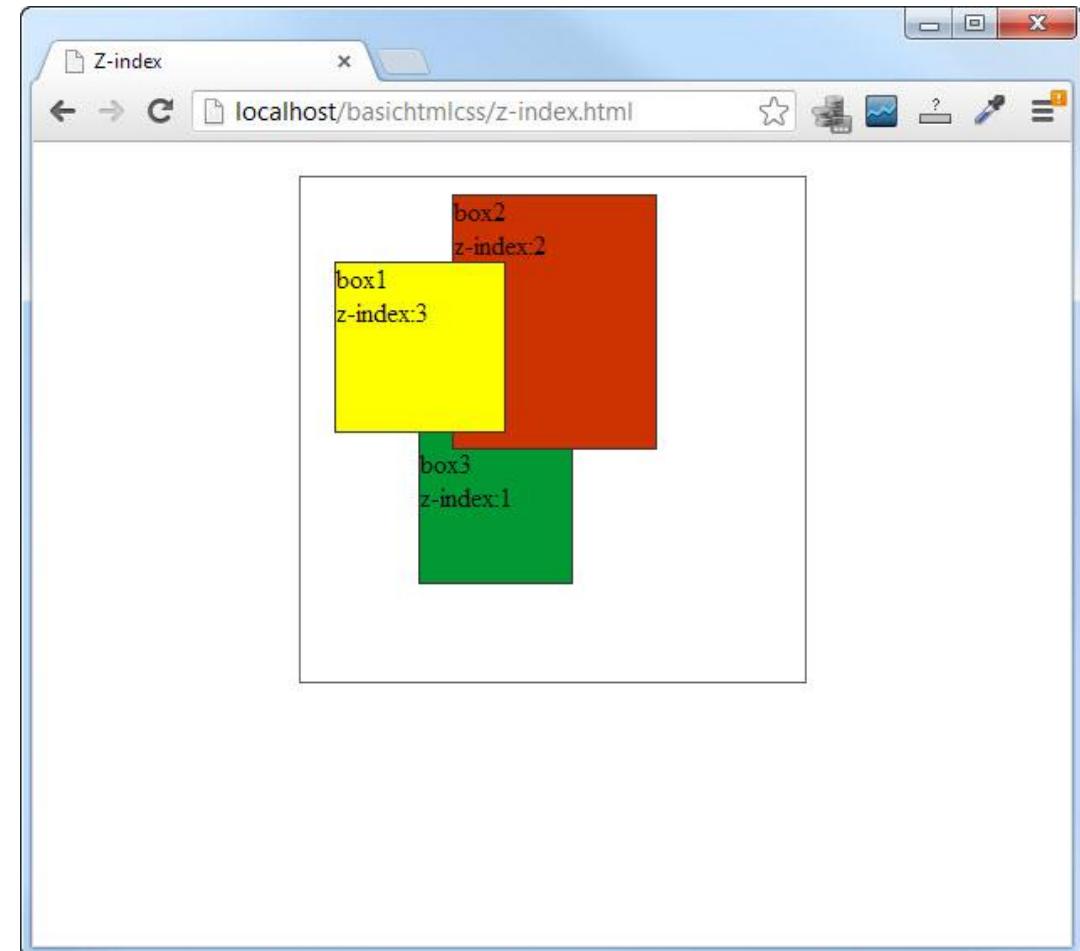
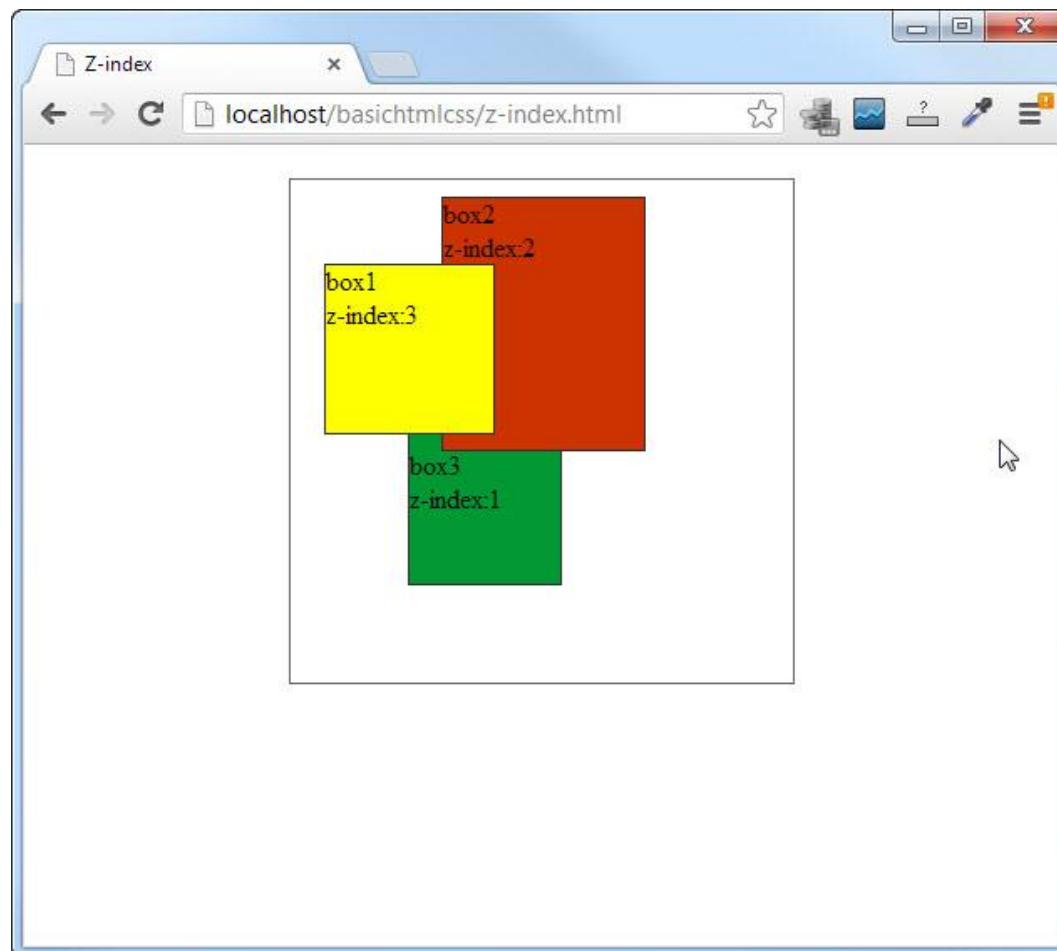
หากไม่มี z-index จะเรียงซ้อนตามการเข้ามา ตัวที่มาหลังจะอยู่ข้างบน เรียงต่อกันไป

HOW TO CHOOSE A  
UNIVERSITY?

Aliquam Sed mauris quis tincidunt Duis tristique  
vitae Vestibulum In consequat. A eget morbi  
consectetuer tincidunt.



# ตัวอย่าง Z-index



# Exercise Z-index

 HIGHLIGHT



ร้าน ยามานะ อาหารญี่ปุ่น  
Yamane Japanese Restaurant

Yamane / อาหารญี่ปุ่นที่สาขาติดกับคนญี่ปุ่น จนเป็นร้านเดียวที่ได้รับเลือกให้เปิดที่โรงงานโตโยต้าที่บ้านฯ เปิดบริการล่า...

[REVIEWS](#)





วอเตอร์ไซด์ รีสอร์ท เรสเตอร์องก์  
Waterside Resort Restaurant

Waterside Resort Restaurant / ถ้ามีก็คงจะเลี้ยงสักรึ หรือมีอีกคนที่เดินทางไปบรรยายกาศสบายนะๆ โรงแรมดีคือ โอบล้อมด้วย...

[REVIEWS](#)





ร้าน ชาลัง อาหารสไตล์เกาหลี  
Salang

Salung สาขาบางจาก / เป็นสาขาใหม่ที่เพิ่งเปิดบริการหลังจากสาขาแรกที่ปทุมธานีรีสอร์ต ราชเทวีได้รับการตอบรับอย่างล้นหลาม ร...

[REVIEWS](#)



 HIGHLIGHT



[REVIEWS](#)

# SYNTAX ໃກ່ ຂອງ HTML5

# SYNTAX ໃກ້ ບວງ HTML5

ສ່ວນຄອນເຖິງ  
ແບ່ງເປັນ 3 ຄວລມນ

ສ່ວນນໍາກາງ  
(navigation)



ສ່ວນຫົວພ້ອມໂລໂກ

ໂນບ່ານ

ສ່ວນ footer

# Syntax ใหม่ในการจัดโครงสร้างของหน้าเว็บ

จากตัวอย่างหน้าเว็บเพจที่แสดงผ่านมา จะถูกแบ่งเป็น

- ส่วนหัวพร้อมโลโก้ และชื่อหัวข้อ
- แถบนำทาง (Navigation bar)
- ส่วนคอนเทนต์ที่ถูกแบ่งเป็นสามคอลัมน์
- บทความ และบล็อกโฆษณาภายในคอลัมน์
- บทความ และบล็อกโฆษณาภายในคอลัมน์
- ส่วนท้ายเพื่อแสดงข้อมูลผู้เขียน และการส่งวันลิขสิทธิ์

มาทำความรู้จัก อิเลเมนต์ใหม่ล่าสุดจาก HTML 5 กัน

# อัลเมนต์ main

```
<main>Text</main>
```

เป็นกลุ่มอัลเมนต์สำหรับจัดการครอบเนื้อหาทั้งหมดในส่วนของ Content เอาไว้เพื่อจัดคุณสมบัติบางอย่าง เช่น จัดความกว้างหน้าเพจ จัดตำแหน่ง กี๊กกลาง เป็นต้น

# อัลเมนต์ header

```
<header>Text</header>
```

เป็นกลุ่มอัลเมนต์สำหรับแนะนำ หรือตัวช่วยนำทาง โดยพื้นฐานการใช้อัลเมนต์นี้ให้เกียบกับการใช้ `<div id="header">` สามารถนำมา header นี้มาแทนได้เลย และไม่จำกดว่าจะใช้กี่ครั้งบนหน้าเพจก็ได้

# อัลเมนต์ section

```
<section>Text</section>
```

เป็นอัลเมนต์ที่ใช้กำหนดส่วนย่อยกันไปของเอกสาร หรือแอปพลิเคชัน การใช้อัลเมนต์นี้ควรเกี่ยวกับความสัมพันธ์ด้วย กล่าวคือควรมีความหมายถึงการแบ่งส่วนเนื้อหา

# อวิลเมนต์ section (ต่อ)

ตัวอย่างแนวทางการเลือกใช้อวิลเมนต์ section นี้

- ใช้แบ่งส่วนย่อยของเนื้อหาที่แสดงเป็นแบบ
- ใช้ในหน้า “เกี่ยวกับเรา” ที่มีการแบ่งเนื้อหาเป็น ประวัติบริษัท, คำแฉลงพันธกิจ, กิจงาน, วิสัยทัศน์ ฯลฯ
- ใช้ในหน้า “ข้อกำหนดการใช้บริการ” ที่มีเนื้อหายาว ๆ และต้องการแบ่งเป็นหัวข้อ
- ใช้ในส่วนย่อยของเว็บไซต์ข่าวออนไลน์ เช่นการใช้ section แบ่งประเภทของข่าวเป็น ข่าวกีฬา, สถานการณ์โลก, ข่าวการเมือง, บันเทิง, เศรษฐกิจ ฯลฯ

The image shows three distinct sections from a news or financial website, each featuring a horizontal navigation bar and a main content area:

- Markets:** Shows stock market data for Asia, Europe, and U.S. It includes a table for Hang Seng, Nikkei, and ASX 100, with a 'More' button.
- Business:** Headlines include "How women cracked tennis' glass ceiling", "U.S. grounds Dreamliners over fire risk", and "Facebook enables free iPhone calls". It has a 'More' button.
- World Sport:** Headlines include "Cycling: Lance stripped of medal", "Football: Guardiola to join Bayern", and "Tennis: Sharapova to face Venus". It has a 'More' button.

Below these sections is a sidebar with a "Daily Snapshot" featuring a horse riding under fire, a "Send your images to iReport" link, and a "More" button. The Nikon logo is also present.

# อิลิเมนต์ article

<article>Text</article>

เป็นอิลิเมนต์ชิ้นเดียวที่สามารถถอยู่ได้โดยลำพังด้วยตัวเอง ไม่จำเป็นต้องมีองค์ประกอบ  
ย่อยร่วมเหมือน section ซึ่งคำแนะนำในการใช้งานเช่น

- เนื้อหาในกระถุก
- บกความในนิตยสารหรือหนังสือพิมพ์
- เนื้อหาในบล็อก
- ความคิดเห็นที่ผู้ใช้พิมพ์เข้ามา

อิลิเมนต์ article สามารถซ้อนเข้าไปภายในอิลิเมนต์ article อื่นได้ หรือสามารถซ้อน  
Section เข้าไปใน article ก็ได้เช่นกัน

## อัลเมนต์ nav

```
<nav>Text</nav>
```

เป็นอัลเมนต์สำหรับการใช้ในการสร้างตัวนำทาง (navigation) สามารถใช้ nav ได้มากกว่าหนึ่งครั้งในหน้าหนึ่ง ๆ ของเว็บเพจ ซึ่ง nav นี้อาจใช้แทนในจุดที่คุณเคยใช้ class="nav" หรือ id="nav" เป็นต้น

## อัลเมนต์ aside

```
<aside>Text</aside>
```

อัลเมนต์นี้มีความหมายว่าเป็นส่วนในหน้าเว็บที่ “มีความสัมพันธ์กับอ้อมกับคอนเทนต์รอบ ๆ” ส่วนใหญ่ใช้เป็น side bar หรือครอบส่วนที่มีคอนเทนต์อย่างและสัมพันธ์กับอัลเมนต์อื่น

# อวิลเมนต์ footer

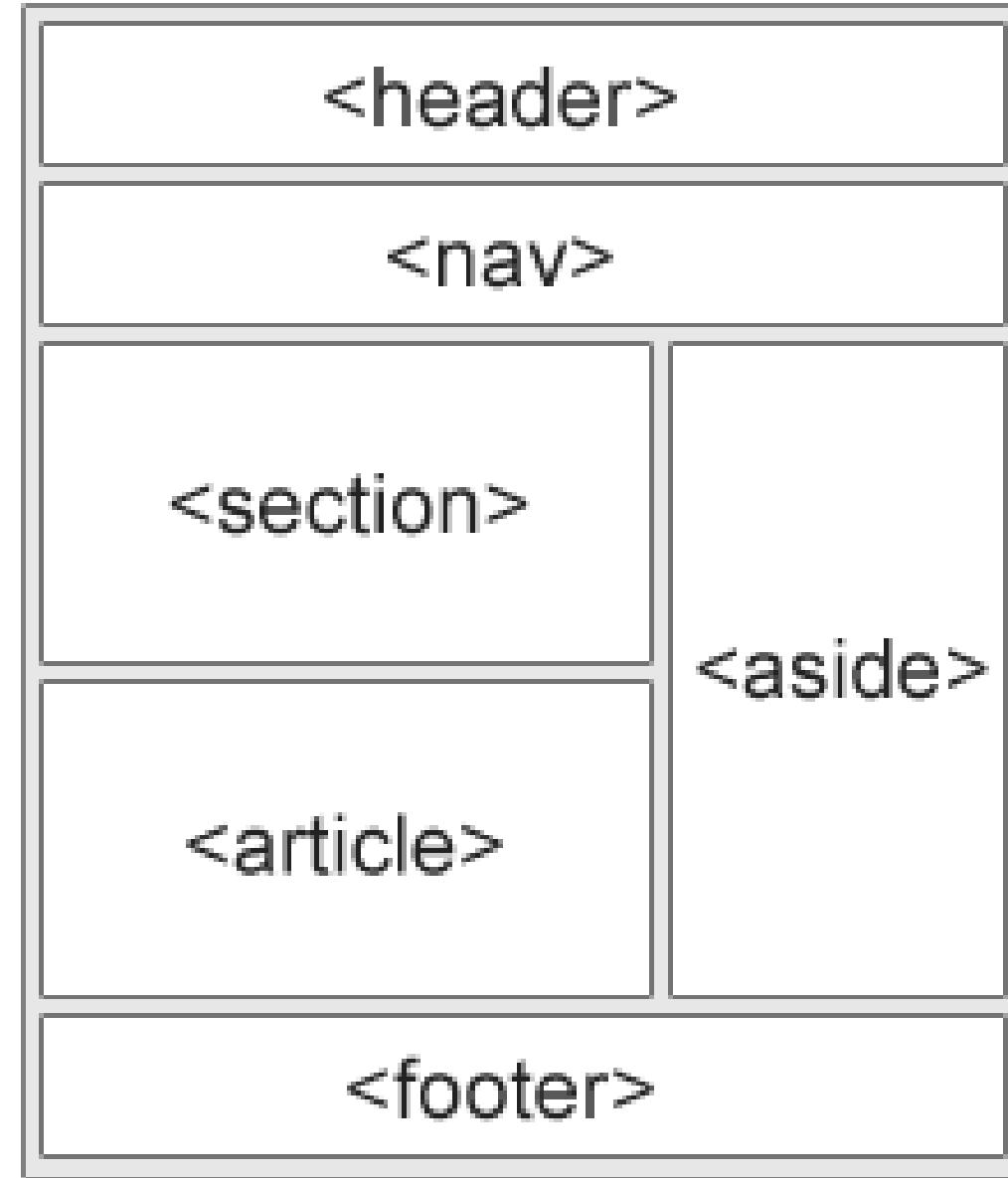
<footer>Text</footer>

อวิลเมนต์สุดท้ายที่จะกล่าวถึงในตอนนี้คือ footer เช่นเดียวกับ header สามารถใช้ได้หลายครั้งในหนึ่งหน้า และสามารถใช้อวิลเมนต์นี้แทนการใช้ <div id="footer"> ได้เลย

a. lacinia vel velit.	tas purus in blandit.	ut nibh.
<b>ALEXIS GOLDSTEIN</b> Maecenas quis tortor arcu. Vivamus rutrum nunc non neque consectetur quis placerat neque lobortis. Nam vestibulum, arcu sodales feugiat consectetur, nisl orci bibendum elit, eu euismod magna sapien ut nibh.	<b>LOUIS LAZARIS</b> Maecenas quis tortor arcu. Vivamus rutrum nunc non neque consectetur quis placerat neque lobortis. Nam vestibulum, arcu sodales feugiat consectetur, nisl orci bibendum elit, eu euismod magna sapien ut nibh.	<b>ESTELLE WEYL</b> Maecenas quis tortor arcu. Vivamus rutrum nunc non neque consectetur quis placerat neque lobortis. Nam vestibulum, arcu sodales feugiat consectetur, nisl orci bibendum elit, eu euismod magna sapien ut nibh.

© SITEPOINT 

# WORKSHOP



# CSS 3

# CSS3 Modules ที่น่าเรียนรู้

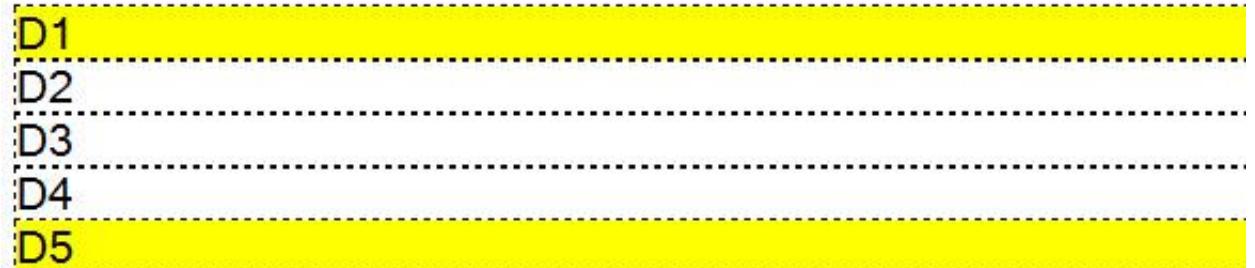
- Selectors
- Box Model
- Backgrounds and Borders
- Text Effects
- fonts
- 2D/3D Transformations
- Animations
- Multiple Column Layout
- User Interface

# Selector

Selector type	Pattern	Description
Substring matching attribute selector	E[att^="val"]	Matches any E element whose att attribute value begins with “val”.

```
div[id^="nav"] { background:#ff0; }
```

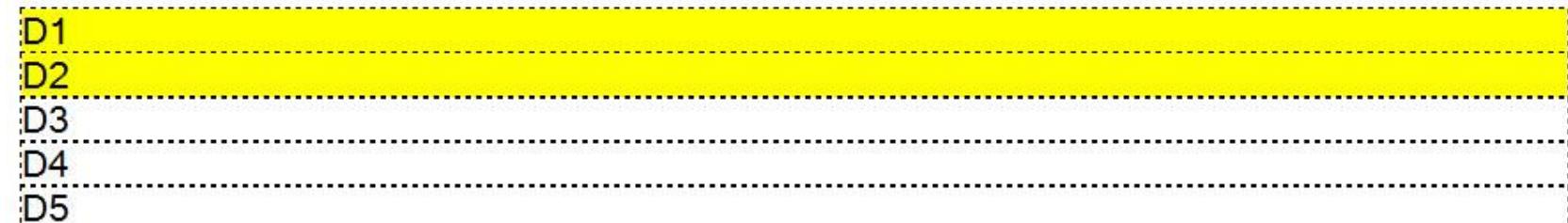
```
<div id="nav-primary">D1</div>
<div id="content-primary">D2</div>
<div id="content-secondary">D3</div>
<div id="tertiary-content">D4</div>
<div id="nav-secondary">D5</div>
```



Selector type	Pattern	Description
Substring matching attribute selector	E[att\$="val"]	Matches any E element whose attribute value ends with “val”.

```
div[id$="primary"] { background:#ff0; }
```

```
<div id="nav-primary">D1</div>
<div id="content-primary">D2</div>
<div id="content-secondary">D3</div>
<div id="tertiary-content">D4</div>
<div id="nav-secondary">D5</div>
```



Selector type	Pattern	Description
Substring matching attribute selector	E[att*="val"]	Matches any E element whose att attribute value contains the substring “val”.

```
div[id*="content"] { background:#ff0; }
```

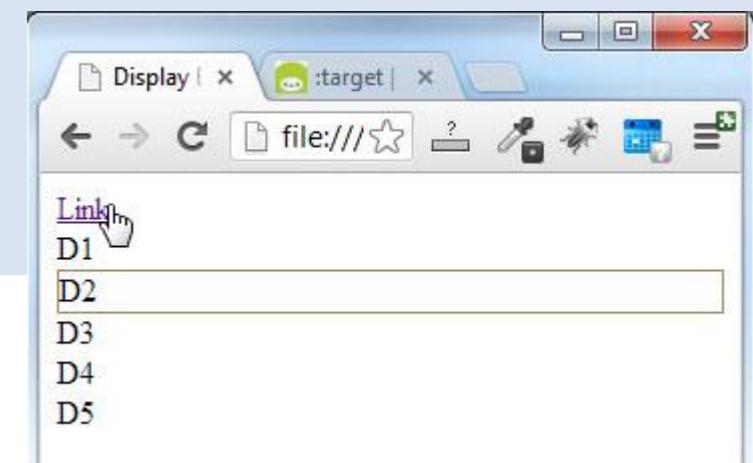
```
<div id="nav-primary">D1</div>
<div id="content-primary">D2</div>
<div id="content-secondary">D3</div>
<div id="tertiary-content">D4</div>
<div id="nav-secondary">D5</div>
```



Selector type	Pattern	Description
Target pseudo-class	E:target	Matches an E element that is the target of the referring URL.

```
div#content-primary:target {border:1px solid #9f8b5e;}
```

```
<a href="#content-primary">Link</a>
<div id="nav-primary">D1</div>
<div id="content-primary">D2</div>
<div id="content-secondary">D3</div>
<div id="tertiary-content">D4</div>
<div id="nav-secondary">D5</div>
```



# Borders

# CSS3 Rounded Corners

```
<style>
div
{
border:2px solid #a1a1a1;
padding:10px 40px;
background:#dddddd;
width:300px;
border-radius:25px;
-moz-border-radius:25px; /* Old Firefox */
}
</style>
```

# CSS3 Rounded Corners

```
<body>
<div>The border-radius property allows you to add rounded corners
to elements.</div>
</body>
```

The border-radius property allows you to  
add rounded corners to elements.

**border-top-left-radius:5px;**  
**border-top-right-radius:10px;**  
**border-bottom-right-radius:15px;**  
**border-bottom-left-radius:40px;**

# CSS3 Box Shadow

```
<style>
div
{
width:300px;
height:100px;
background-color:yellow;
-moz-box-shadow: 10px 10px 10px 5px #888888;
/* Old Firefox */
box-shadow: 10px 10px 10px 5px #888888;
}
</style>
```

**ค่าแรกคือ** ค่าความเหลื่อมในแนวนอน ค่าที่เป็นบวกจะสร้างเขตกล้ามด้านขวาของอวัลเมนต์ ส่วนค่าที่เป็นลบจะตกไปทางซ้าย

**ค่าที่สองคือ** ค่าความเหลื่อมในแนวตั้ง ค่าที่เป็นบวกจะสร้างเขตกลลงไป ทำให้เกิดเขตทางด้านล่างของอวัลเมนต์ ส่วนค่าที่เป็นลบจะดันเวลาขึ้นไปด้านบน

**ค่าที่สาม (ถ้ามี)** คือ ระดับความพลุ่มวูบของเงา ยิ่งค่านี้มากก็จะยิ่งพร่ามัวมากยิ่งขึ้นและเป็นไปเพียงค่าบวกเท่านั้น เงาของเรานั้นไม่มีความพร่ามัว ดังนั้น เราสามารถกำหนดค่าเป็นศูนย์ (0) หรือจะไว้ไม่ใส่ค่าก็ช่วยกัน

**ค่าที่สี่คือ** ระยะการแผ่กระจายของเงา ค่าบวกจะทำให้เงาแผ่กระจายไปทุกทิศทาง ค่าลบจะทำให้เงาลดขนาดลง เงาของเรานั้นไม่แผ่กระจาย ดังนั้น เราสามารถกำหนดค่าเป็นศูนย์ (0) หรือจะไว้ไม่ใส่ค่าก็ได้ช่วยกัน

# CSS3 Text Effects

# CSS3 Backgrounds Size

```
<style>
h1
{
text-shadow: 5px 5px 5px #FF0000;
}
</style>
```

```
<body>
<h1>Text-shadow effect!</h1>
</body>
```

**Text-shadow effect!**

# CSS3 Word Wrapping

If a word is too long to fit within an area, it expands outside:

This paragraph contains a very long word:  
thisisaveryveryveryveryverylongword.  
The long word will break and wrap to the next line.

In CSS3, the word-wrap property allows you to force the text to wrap - even if it means splitting it in the middle of a word:

This paragraph contains a very long word:  
thisisaveryveryveryveryveryverylongword. The long word will break and wrap to the next line.

# CSS3 Word Wrapping

```
<style>
p.test
{
width:11em;
border:1px solid #000000;
word-wrap:break-word;
}
</style>
```

# CSS3 Word Wrapping

```
<body>
<p class="test"> This paragraph contains a very long word:  
thisisaveryveryveryveryveryverylongword. The long word will break and wrap to  
the next line.</p>
</body>
```

This paragraph contains  
a very long word:  
thisisaveryveryveryveryv  
eryverylongword. The  
long word will break and  
wrap to the next line.

# **CSS3 Fonts**

```
<style>
@font-face
{
font-family: myFirstFont;
src: url('supermarket.ttf') ,
     url('supermarket.eot'); /* IE9+ */
}
div
{
font-family:myFirstFont;
font-size:30px;
}
</style>
```

# **CSS3 2D Transforms**

# 2D Transforms

In this chapter you will learn about the 2d transform methods:

- `translate()`
- `rotate()`
- `scale()`
- `skew()`
- `matrix()`

## 2D Transforms

```
<style>
div
{
width:200px;
height:100px;
margin-top:50px;
background-color:yellow;

/* Rotate div */
transform:rotate(30deg);
-ms-transform:rotate(30deg); /* IE 9 */
-moz-transform:rotate(30deg); /* Firefox */
-webkit-transform:rotate(30deg); /* Safari and Chrome */
-o-transform:rotate(30deg); /* Opera */
}
</style>
```

# **CSS3 Animation**

# 2D Animation

```
@keyframes myanimation{  
    0: left  
    30: right  
    60: top  
    120:bottom  
}
```

# Responsive Design

The image shows three separate Mozilla Firefox browser windows side-by-side, all displaying different parts of the Microsoft Home Page.

- Left Window:** Shows the main Microsoft homepage featuring the Surface Pro tablet. It includes a large orange callout box with the text "Surface Pro" and "A powerful PC in tablet form". Below it, there's a "Discover" section with links to Windows, Office, Surface, and Windows Phone.
- Middle Window:** Shows a specific page for "Devices and Services". It features a purple background with a cartoon illustration of people working on a cloud-based puzzle. Below the illustration, there's a section for "Visual Studio" with the text "Amazing apps start with Visual Studio" and a link to "Download the Visual Studio 2013 Preview".
- Right Window:** Shows another part of the Microsoft Home Page. It includes a search bar for "Find a Microsoft Store near you". Below the search bar, there's a "Home year of" section showing Microsoft products like Office 365 and XBOX. At the bottom, there are links for Surface, Windows Phone, Xbox, Skype, and Bing.

# ความหมาย และ ความสำคัญ ของ Responsive Web Design

ในปัจจุบัน Mobile Internet Users ได้มีจำนวนเพิ่มขึ้นอย่างรวดเร็ว และมีแนวโน้มที่จะแซง Desktop Internet Users ในปี 2013 อีกด้วย ซึ่ง Mobile Devices นั้นมีความหลากหลายมาก ไม่ว่าจะเป็น ขนาดและความละเอียดของหน้าจอแสดงผล(screen size and resolution) แนวของการแสดงผล(orientation) หรือแม้แต่ระบบปฏิบัติการ(OS)

ถ้าเป็นสมัยก่อน เราต้องทำเว็บไซต์ออกมาหลายๆ version เช่น Desktop version กับ Mobile version เพื่อให้เว็บไซต์ของเรา สามารถแสดงผลได้อย่างเหมาะสมกับ Device นั้นๆ ซึ่งวิธีนี้จะทำให้ต้นทุนเพิ่มขึ้น ทั้งในด้านเวลาและค่าจ้างในการพัฒนา

Responsive Web Design คือ การออกแบบเว็บไซต์ด้วยแนวคิดใหม่ ที่จะทำให้เว็บไซต์สามารถแสดงผลได้อย่างเหมาะสม บนอุปกรณ์ที่แตกต่างกัน โดยใช้โค้ดร่วมกัน URL เดียว กับ เพื่อแก้ปัญหาดังกล่าว

# หลักการของ Responsive Web Design

การจะทำ Responsive Web Design มักใช้เทคนิคหลายๆ อย่าง ร่วมกัน ไม่ว่าจะเป็น Fluid Grid, Flexible Images และ CSS3 Media Queries

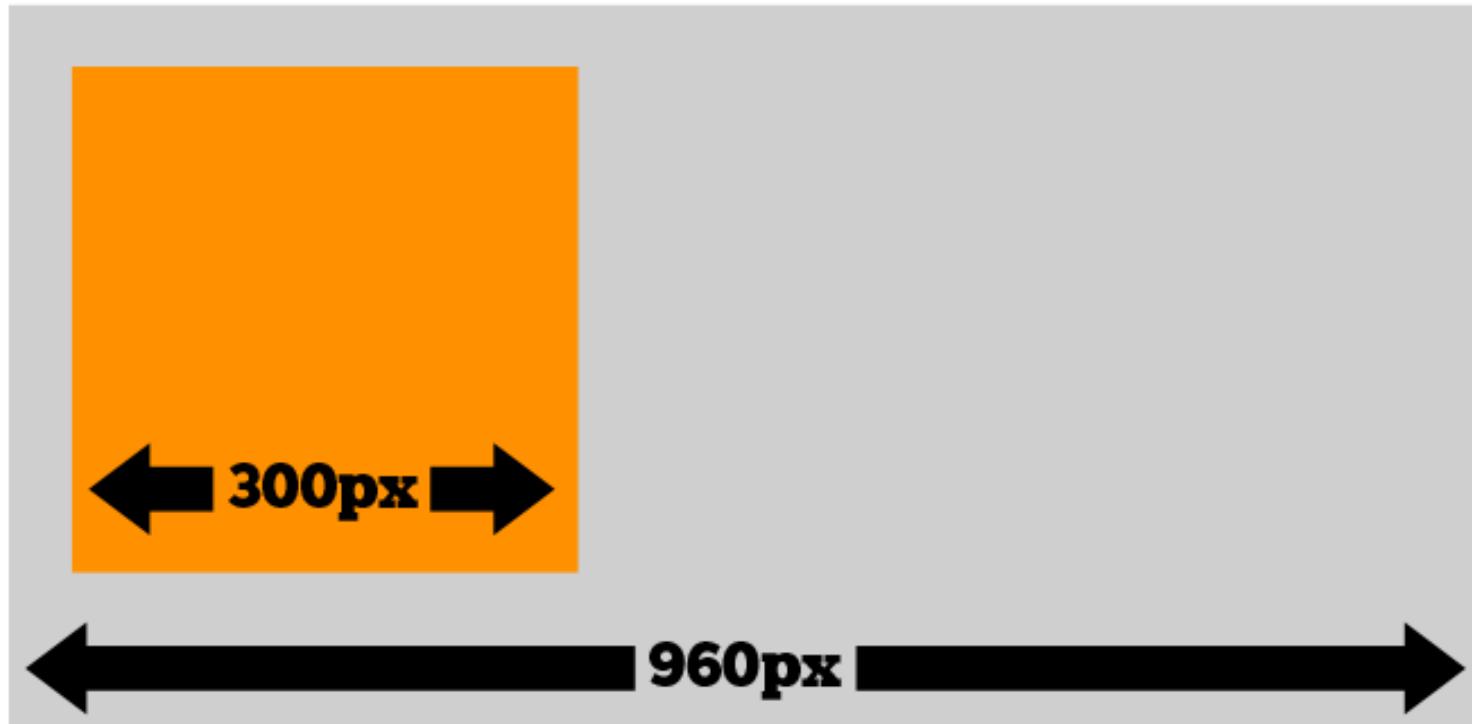
เริ่มแรกคือการทำ Fluid Grid ซึ่งก็คือการอวอกแบบ Grid ให้เป็นแบบ Relative ซึ่งก็คือการที่ไม่ได้กำหนดขนาดของ Grid แบบตายตัว แต่จะกำหนดให้สัมพันธ์กับสิ่งอื่นๆ เช่น กำหนดความกว้างแบบเป็น % หรือการใช้ font-size หน่วยเป็น em เป็นต้น

ต่อมาคือการทำ Flexible Images หรือการกำหนดขนาดของ Images ต่างๆ ให้บีความสัมพันธ์กับขนาดของหน้าจอแสดงผล หากรูปต้นฉบับมีขนาดใหญ่มาก เวลาแสดงในมือถือที่มีจอขนาดเล็ก ก็ควรลดขนาดลงมา เพื่อให้แสดงผลได้อย่างสวยงาม เป็นต้น

สุดท้ายคือการใช้ CSS3 Media Queries ซึ่งจะช่วยให้เราสามารถกำหนด style sheets สำหรับ Devices ต่างๆ ได้โดยส่วนใหญ่ เราจะเขียน style sheets พื้นฐานเอาไว้ ซึ่งกลุ่มนี้ จะไม่ขึ้นอยู่กับ Devices ใดๆ หลังจากนั้นให้เราเขียน style sheets สำหรับ Devices ที่มีขนาดหน้าจอที่เล็กสุด เพิ่มขึ้นไปเรื่อยๆ จนถึงขนาดใหญ่สุด ซึ่งการเขียนแบบนี้ จะช่วยลดความซ้ำซ้อนของโค้ด และยังทำให้การแก้โค้ดในภายหลังทำได้ง่ายอีกด้วย

# Fluid Grids

# **target / context = result**



$$\mathbf{300px} \ / \ \mathbf{960px} = \mathbf{31.25\%}$$

**\*These measurements are not to scale.**

# Flexible Media

The final, equally important aspect to responsive web design involves flexible media. As viewports begin to change size media doesn't always follow suit. Images, videos, and other media types need to be scalable, changing their size as the size of the viewport changes.

One quick way to make media scalable is by using the max-width property with a value of 100%. Doing so ensures that as the viewport gets smaller any media will scale down according to its containers width.

```
img, video, canvas,iframe {  
    max-width: 100%;  
    width:100%;  
}
```

# CSS3 Media Queries

# ตัวอย่างการกำหนด Media Queries

```
@media only screen and (max-width: 480px)
{
    /* Styles */
}
```

จากตัวอย่างด้านบน Styles จะถูกเรียกใช้หาก Device นั้นๆ มีความกว้างของหน้าจอไม่เกิน 480px

# ຕັ້ງອໝາງການກຳນົດ Media Queries (Desktop first)

```
@media screen and (max-width: 1170px) {  
    /* Add your styles for devices with a maximum width of 1170 */  
}  
  
@media screen and (max-width: 768px) {  
    /* Add your styles for devices with a maximum width of 768 */  
}  
  
@media screen and (max-width: 320px) {  
    /* Add your styles for devices with a maximum width of 320 */  
}
```

# ตัวอย่างการกำหนด Media Queries (Mobile first)

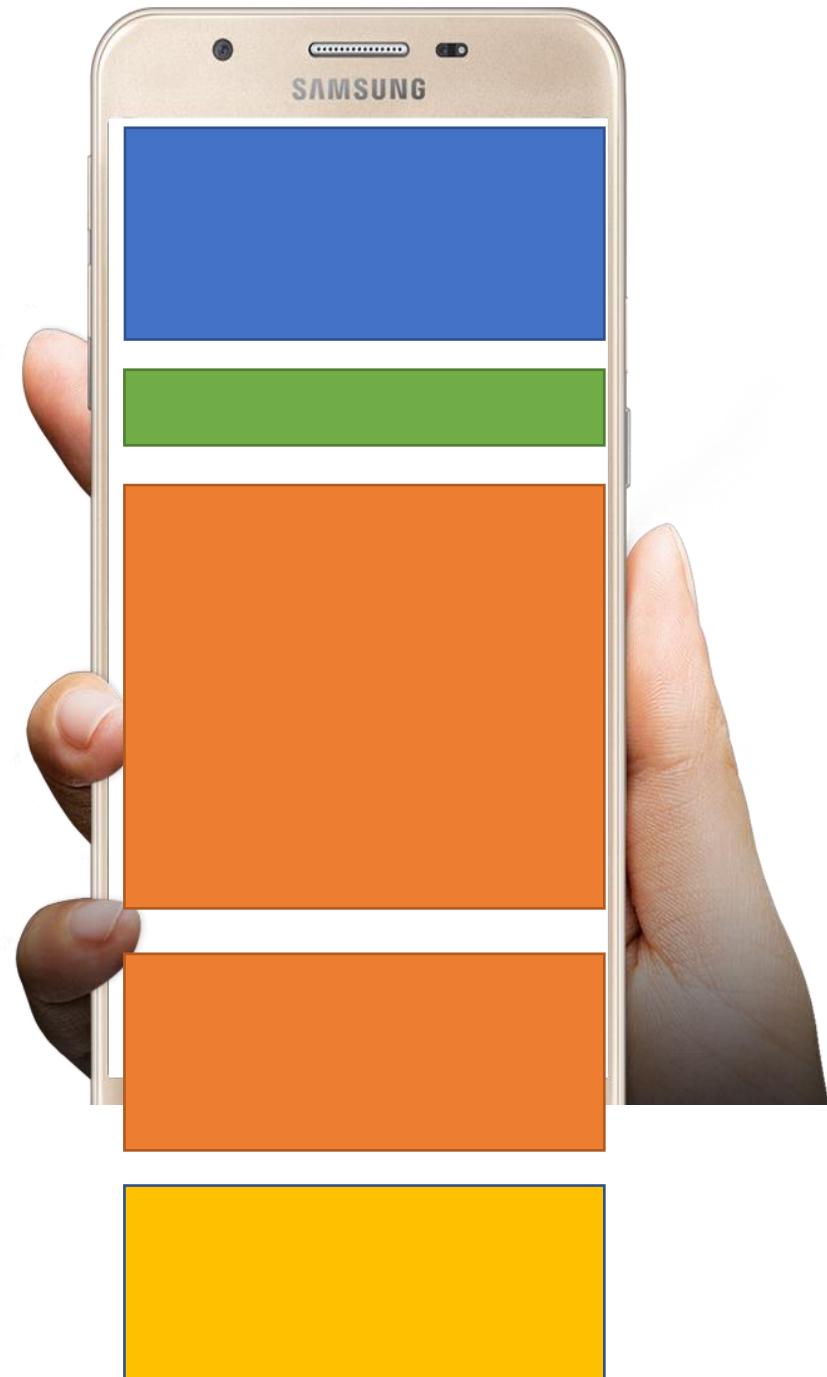
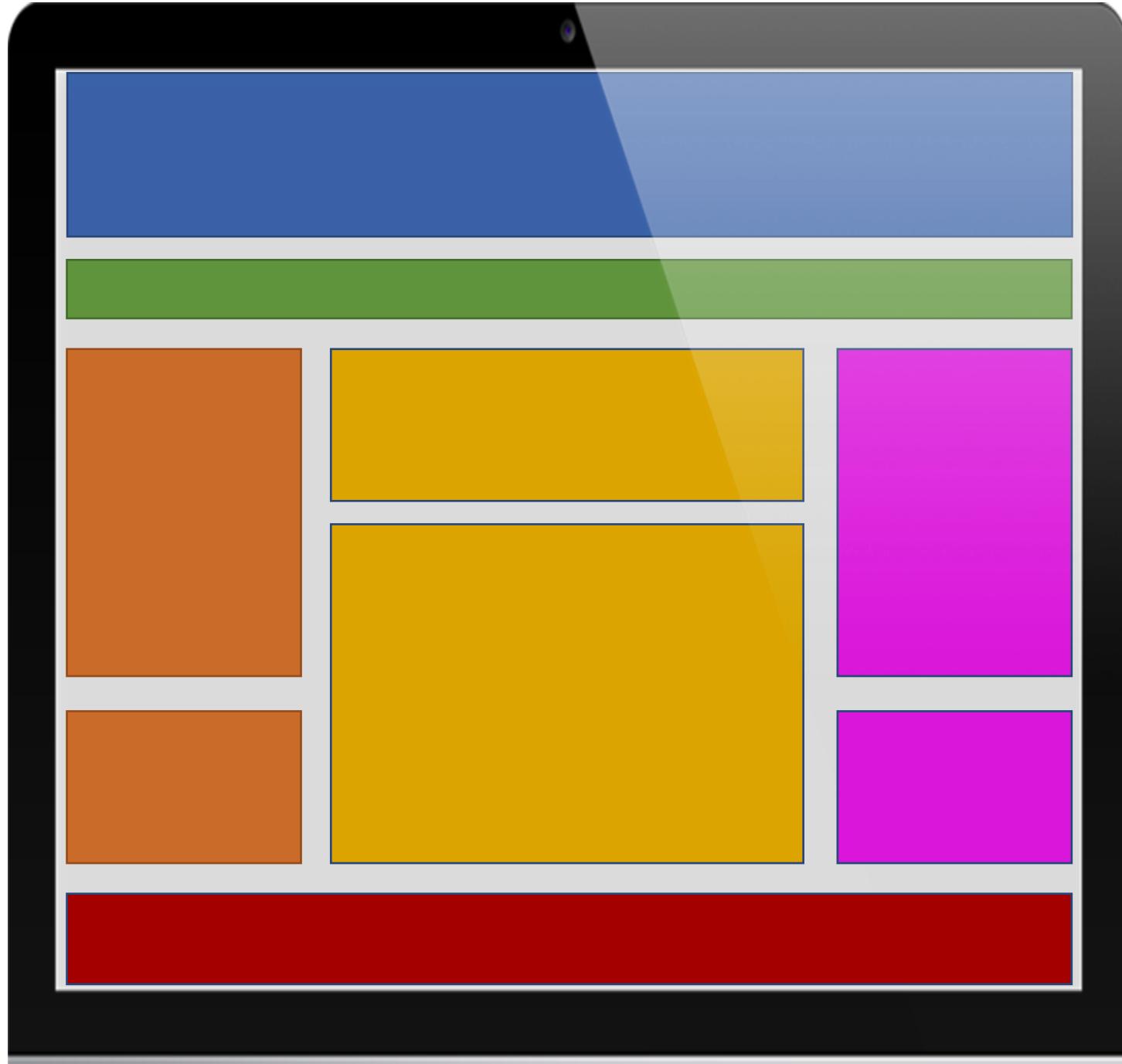
```
@media screen and (min-width: 320px) {  
    /* Add your styles for devices with a minimum width of 320 */  
}  
  
@media screen and (min-width : 768px) {  
    /* Add your styles for devices with a minimum width of 768 */  
}  
  
@media screen and (min-width : 1170px) {  
    /* Add your styles for devices with a minimum width of 1170 */  
}
```

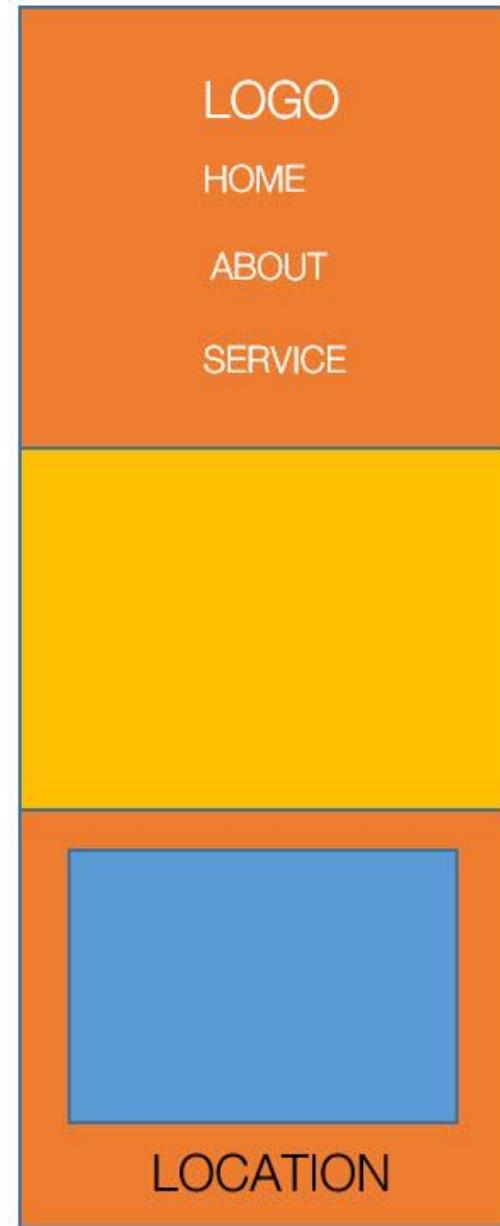
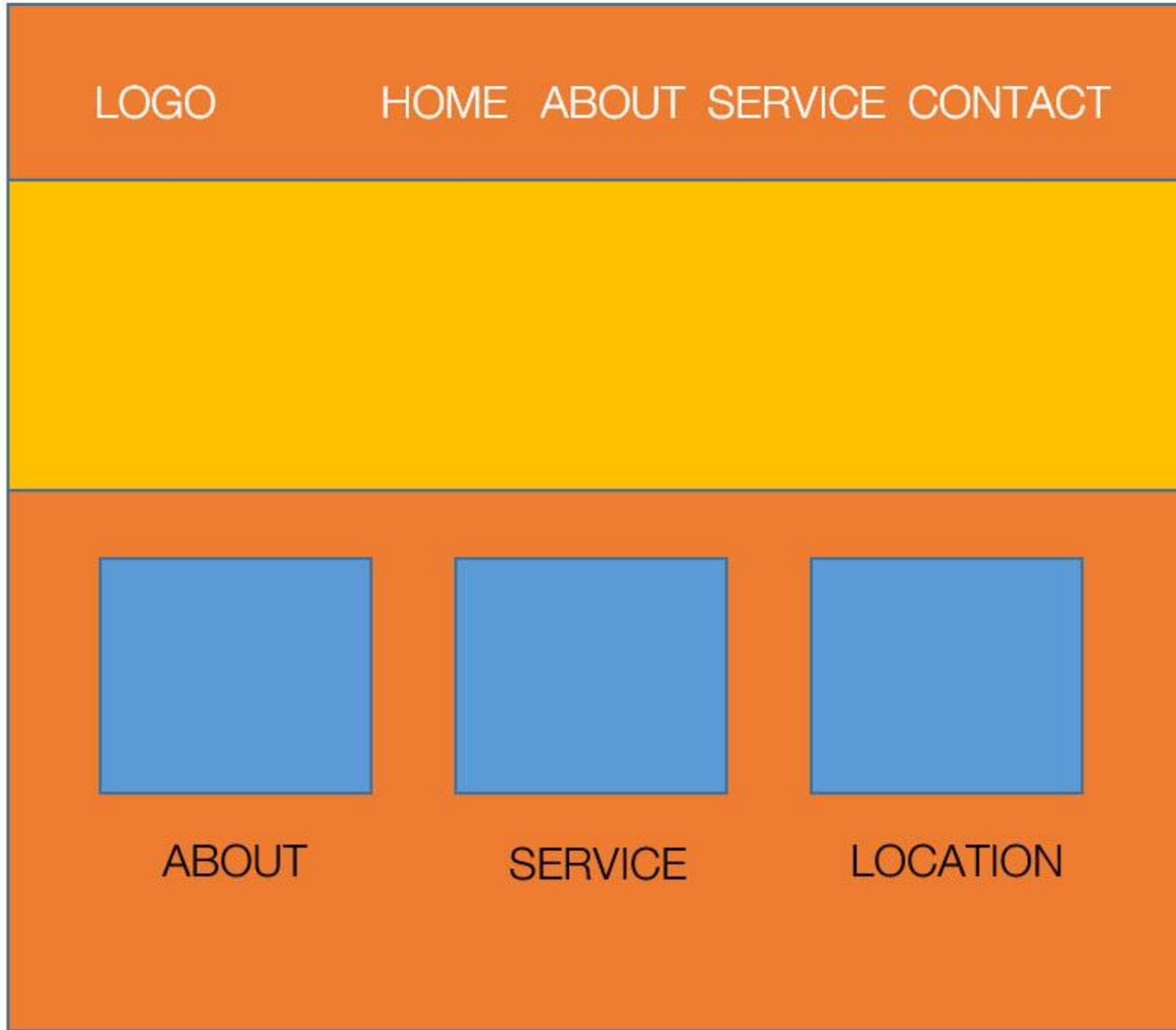
# Viewport Scale

```
<meta name="viewport" content="width=device-width, initial-scale=1.0" />
```

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <meta http-equiv="X-UA-Compatible" content="ie=edge">
7
8      <title>Show Products</title>
9  </head>
10 <body>
```

# Exercise





# JavaScript คืออะไร ?



## Most Popular Technologies

### Programming, Scripting, and Markup Languages



**Front-end Developer**

**Back-end Developer**

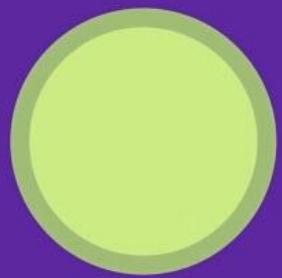
**Full-stack Developer**



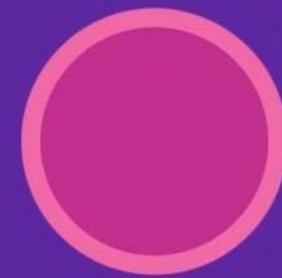
Web / Mobile  
Apps



Real-time  
Networking  
Apps

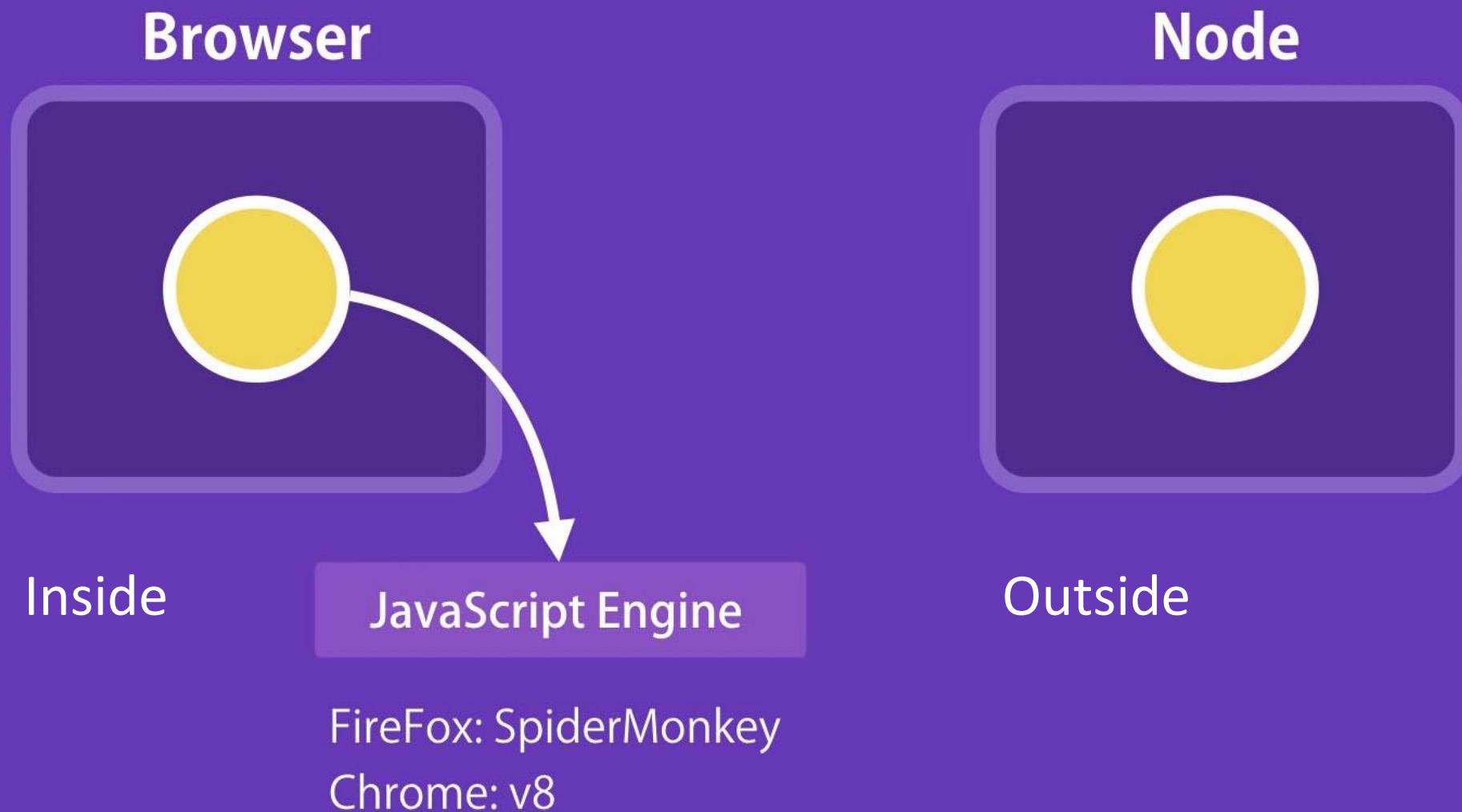


Command-line  
Tools



Games

# JavaScript ทำงานอย่างไร



# JavaScript vs ECMAScript

# ECMAScript

Specification



กำหนดมาตรฐาน

# JavaScript

Programming Language



ตัวภาษาที่กำหนดมาตรฐาน

v1

ES2015/ES6



# บริษัท เอคมา อินเตอร์เนชันแนล

บทความ พูดคุย

จากวิศว์เดียว สารานุกรมเสรี

"Ecma" เปเปลี่ยนเส้นทางมาที่นี่ สำหรับการใช้งานอื่น ๆ โปรดอ่าน ECMA

**Ecma International** ( /'ɛkmoʊ/ ) เป็นองค์กรมาตรฐานที่ไม่แสวงหากำไร สำหรับระบบสารสนเทศและการสื่อสาร<sup>[1]</sup> องค์กรได้รับข้อบัญญัติในปี 1994 เมื่อ European Computer Manufacturers Association (ECMA) เปเปลี่ยนชื่อเพื่อสะท้อนถึงการเข้าถึงและกิจกรรมระดับโลกขององค์กร ด้วยเหตุนี้ ชื่อจึงไม่ถือเป็นตัวย่ออีกด้อไป และไม่ใช่ตัวพิมพ์ใหญ่เต็มตัวอีกด้อไป องค์กรนี้ก่อตั้งขึ้นในปี พ.ศ. 2504 เพื่อสร้างมาตรฐานระบบคอมพิวเตอร์ในยุโรป สมาคมภาพเปิดกว้างสำหรับบริษัทขนาดใหญ่และขนาดเล็กทั่วโลกที่ผลิต ทำการตลาด หรือพัฒนาระบบคอมพิวเตอร์หรือการสื่อสาร และมีความสนใจและประสบการณ์ในสาขาที่หน่วยงานด้านเทคโนโลยีดังกล่าว องค์กรนี้ตั้งอยู่ในเจนีวา

## จุดมุ่งหมาย [ แก้ไข ]

Ecma มุ่งมั่นที่จะพัฒนามาตรฐานและรายงานทางเทคโนโลยีเพื่ออำนวยความสะดวกและกำหนด มาตรฐานการใช้เทคโนโลยีสารสนเทศและการสื่อสารและอุปกรณ์เวิล์ฟลิกทรอนิกส์สำหรับผู้ใช้ ส่งเสริมการใช้มาตรฐานอย่างถูกต้องโดยมีอิทธิพลต่อสภาพแวดล้อมที่นำไปใช้ และเผยแพร่ มาตรฐานและรายงานเหล่านี้ในรูปแบบอิเล็กทรอนิกส์และสิ่งพิมพ์ สิ่งพิมพ์ของ Ecma รวมถึง มาตรฐาน สามารถคัดลอกได้โดยอิสระโดยผู้ที่สนใจใช้ทั้งหมดโดยไม่มีข้อจำกัดด้านลิขสิทธิ์ การ พัฒนามาตรฐานและรายงานทางเทคโนโลยีจะดำเนินการร่วมกับองค์กรระดับชาติ ระดับยุโรป และระดับนานาชาติที่เกี่ยวข้อง

ต่างจากหน่วยงานมาตรฐานแห่งชาติ Ecma เป็นองค์กรที่เน้นมาตรฐานหลัก โดยมีความภูมิใจใน แนวทาง "เชิงธุรกิจ" ที่ได้มาจากมาตรฐาน ซึ่งอ้างว่าช่วยให้ได้มาตรฐานที่ดีขึ้นในเวลาอันสั้นกว่า เนื่องจากกระบวนการราชการน้อยลงที่เน้นการรอลุลอดโดยจันทร์ <sup>[2]</sup> [ จันทร์ต้องล้างจึง ]

Ecma มีส่วนสนับสนุนอย่างแท้จริงในการสร้างมาตรฐานด้านเทคโนโลยีสารสนเทศและ โทรคมนาคมทั่วโลก มาตรฐาน Ecma มากกว่า 400 มาตรฐาน<sup>[2]</sup> และรายงานทางเทคนิค 100 รายงาน<sup>[3]</sup> ได้รับการเผยแพร่ ซึ่งมากกว่า <sup>2</sup> ใน <sup>3</sup> ของ มาตรฐานเหล่านี้ได้รับการนำมาใช้เป็นมาตรฐานสากลและ/หรือรายงานทางเทคโนโลยีด้วย

อ่าน แก้ไข คุ้มครอง เครื่องมือ

## บริษัท เอคมา อินเตอร์เนชันแนล



การก่อตัว	1961 ; 63 ปีที่ผ่านมา
พิมพ์	การจัดมาตรฐานองค์กร
วัดกุประสงค์	การสร้างมาตรฐานเทคโนโลยีสารสนเทศและการสื่อสารและอุปกรณ์ที่รองรับผู้ใช้งาน
สำนักงาน	เจนีวาประเทศไทยและเวลล์ไซด์
ในญี่ปุ่น	ท่าอากาศยานนานาชาติ
พื้นที่ให้บริการ	ท่าอากาศยานนานาชาติ
ภาษา	ภาษาอังกฤษ
หางาน	ภาษาอังกฤษ
บุคคลสำคัญ	โจเซฟ ฟรีดิช (ประธาน) ไมเคิล ชานอฟฟ์ (ประธานคณะกรรมการบริหาร) ชาเมเน่ ยุสเซน (เลขานุการ)
อัจฉริยะหลัก	สมัยใหม่
งบประมาณ	โดยเฉลี่ยประมาณ 100 ล้านดอลลาร์
เว็บไซต์	<a href="http://www.ecma-international.org">www.ecma-international.org</a>
เดิมเรียกว่า	สมาคมผู้ผลิตคอมพิวเตอร์แห่งยุโรป

# ECMA-262, 15<sup>th</sup> edition, June 2024 ECMAScript® 2024 Language Specification



## About this Specification

The document at <https://tc39.es/ecma262/> is the most accurate and up-to-date ECMAScript specification. It contains the content of proposals (those that have reached Stage 4 in the proposal process) and thus are implemented in several implementations and will be taken.

This document is available as a [single page](#) and as [multiple pages](#).

## Contributing to this Specification

This specification is developed on GitHub with the help of the ECMAScript community. There are a number of ways to contribute to

ECMAScript is a JavaScript standard developed by Ecma International. Since 2015, major versions have been published annually. ECMAScript 2024, the 15th and current version, was released in June 2024.

## Versions [edit]

Edition	Date published	Name	Changes from prior edition
1	June 1997		First edition based on JavaScript 1.1 as implemented in Netscape Navigator 3.0. <sup>[1]</sup>
2	June 1998		Editorial changes to keep the specification fully aligned with ISO/IEC 16262:1998.
3	December 1999		Based on JavaScript 1.2 as implemented in Netscape Navigator 4.0. <sup>[2]</sup> Added regular expressions, better string handling, new control statements, try/catch exception handling, tighter definition of errors, formatting for numeric output, and other enhancements
4	Abandoned (last draft 30 June 2003)	ECMAScript 4 (ES4)	Fourth Edition was abandoned, due to political differences concerning language complexity. Many features proposed for the Fourth Edition have been completely dropped; some were incorporated into the sixth edition.
5	December 2009		Adds "strict mode", a subset intended to provide more thorough error checking and avoid error-prone constructs. Clarifies many ambiguities in the 3rd edition specification, and accommodates behavior of real-world implementations that differed consistently from that specification. Adds some new features, such as getters and setters, library support for JSON, and more complete reflection on object properties. <sup>[3]</sup>
5.1	June 2011		Changes to keep the specification fully aligned with ISO/IEC 16262:2011.
6	June 2015 <sup>[4]</sup>	ECMAScript 2015 (ES2015)	See #6th Edition – ECMAScript 2015

7	June 2016 <sup>[5]</sup>	ECMAScript 2016 (ES2016)	See #7th Edition – ECMAScript 2016	Brian Terlson
8	June 2017 <sup>[6]</sup>	ECMAScript 2017 (ES2017)	See #8th Edition – ECMAScript 2017	Brian Terlson
9	June 2018 <sup>[7]</sup>	ECMAScript 2018 (ES2018)	See #9th Edition – ECMAScript 2018	Brian Terlson
10	June 2019 <sup>[8]</sup>	ECMAScript 2019 (ES2019)	See #10th Edition – ECMAScript 2019	Brian Terlson, Bradley Farias, Jordan Harband
11	June 2020 <sup>[9]</sup>	ECMAScript 2020 (ES2020)	See #11th Edition – ECMAScript 2020	Jordan Harband, Kevin Smith
12	June 2021 <sup>[10]</sup>	ECMAScript 2021 (ES2021)	See #12th Edition – ECMAScript 2021	Jordan Harband, Shu-yu Guo, Michael Ficarra, Kevin Gibbons
13	June 2022 <sup>[11]</sup>	ECMAScript 2022 (ES2022)	See #13th Edition – ECMAScript 2022	Shu-yu Guo, Michael Ficarra, Kevin Gibbons
14	June 2023 <sup>[12]</sup>	ECMAScript 2023 (ES2023)	See #14th Edition – ECMAScript 2023	Shu-yu Guo, Michael Ficarra, Kevin Gibbons
15	June 2024 <sup>[13]</sup>	ECMAScript 2024 (ES2024)	See #15th Edition – ECMAScript 2024	Shu-yu Guo, Michael Ficarra, Kevin Gibbons
16	(pending)	ECMAScript 2025 (ES2025)	Pending, see features being considered: #ES.Next	(pending)

## 6th Edition – ECMAScript 2015 [ edit ]

The 6th edition, ECMAScript 6 (ES6) and later renamed to ECMAScript 2015, was finalized in June 2015.<sup>[4][30]</sup> This update adds significant new syntax for writing complex applications, including class declarations (`class Foo { ... }`), ES6 modules like

`import * as moduleName from "..."; export const Foo`, but defines them semantically in the same terms as ECMAScript 5 strict mode. Other new features include iterators and `for...of` loops, Python-style generators, arrow function expression (`() => {...}`), `let` keyword for local declarations, `const` keyword for constant local declarations, binary data, typed arrays, new collections (maps, sets and WeakMap), promises, number and math enhancements, reflection, proxies (metaprogramming for virtual objects and wrappers) and template literals using backticks (`` ``) for multi-line strings without escape characters.<sup>[31][32]</sup> The complete list is extensive.<sup>[33][34]</sup> As the first "ECMAScript Harmony" specification, it is also known as "ES6 Harmony".

## 7th Edition – ECMAScript 2016 [ edit ]

The 7th edition, or ECMAScript 2016, was finalized in June 2016.<sup>[5]</sup> Its features include exponentiation operator `**` for numbers, `await`, `async` keywords for asynchronous programming (as a preparation for ES2017), and the `Array.prototype.includes` function.<sup>[5]</sup> The exponentiation operator is equivalent to `Math.pow`, but provides a simpler syntax similar to languages like Python, F#, Perl, and Ruby. `async` / `await` was hailed as an easier way to use promises and develop asynchronous code.

## 8th Edition – ECMAScript 2017 [ edit ]

The 8th edition, or ECMAScript 2017, was finalized in June 2017.<sup>[6]</sup> Its features include the `Object.values`, `Object.entries` and `Object.getOwnPropertyDescriptors` functions for easy manipulation of Objects, `async` / `await` constructions that use generators and promises, and additional features for concurrency and `atomics`. It also includes `String.prototype.padStart()`.<sup>[35][6]</sup>

## 9th Edition – ECMAScript 2018 [edit]

The 9th edition, or ECMAScript 2018, was finalized in June 2018.<sup>[7]</sup> New features include the spread operator and rest parameters (`...`) for object literals, asynchronous iteration, `Promise.prototype.finally` and additions to RegExp.<sup>[7]</sup>

The spread operator allows for the easy copying of object properties, as shown below.

```
let object = {a: 1, b: 2}

let objectClone = Object.assign({}, object) // before ES2018
let objectClone = {...object} // ES2018 syntax

let otherObject = {c: 3, ...object}
console.log(otherObject) // -> {c: 3, a: 1, b: 2}
```

## 10th Edition – ECMAScript 2019 [edit]

The 10th edition, or ECMAScript 2019, was published in June 2019.<sup>[8]</sup> Added features include, but are not limited to, `Array.prototype.flat`, `Array.prototype.flatMap`, changes to `Array.sort`, and `Object.fromEntries`.<sup>[8]</sup>

`Array.sort` is now guaranteed to be [stable](#), meaning that elements with equal sorting keys will not change relative order before and after the sort operation. `Array.prototype.flat(depth=1)` flattens an array to a specified depth, meaning that all subarray elements (up to the specified depth) are concatenated recursively.

Another notable change is that so-called *catch binding* became optional.<sup>[36]</sup>

## 11th Edition – ECMAScript 2020 [edit]

The 11th edition, or ECMAScript 2020, was published in June 2020.<sup>[9]</sup> In addition to new functions, this version introduces a `BigInt` primitive type for arbitrary-sized integers, the [nullish coalescing operator](#), and the [globalThis object](#).<sup>[9]</sup>

BigInts are created either with the `BigInt` constructor or with the syntax `10n`, where "n" is placed after the number literal. BigInts allow the representation and manipulation of integers beyond `Number.MAX_SAFE_INTEGER`, while Numbers are represented by a double-precision 64-bit IEEE 754 value. The built-in functions in `Math` are not compatible with BigInts; for example, exponentiation of BigInts must be done with the `**` operator instead of `Math.pow`.

The nullish coalescing operator, `??`, returns its right-hand side operand when its left-hand side is `null` or `undefined`. This contrasts with the `||` operator, which would return `"string"` for all "falsy" values, such as the ones below.

## 11th Edition – ECMAScript 2020 [edit]

The 11th edition, or ECMAScript 2020, was published in June 2020.<sup>[9]</sup> In addition to new functions, this version introduces a `BigInt` primitive type for arbitrary-sized integers, the `nullish coalescing operator`, and the `globalThis` object.<sup>[9]</sup>

BigInts are created either with the `BigInt` constructor or with the syntax `10n`, where "n" is placed after the number literal. BigInts allow the representation and manipulation of integers beyond `Number.MAX_SAFE_INTEGER`, while Numbers are represented by a double-precision 64-bit IEEE 754 value. The built-in functions in `Math` are not compatible with BigInts; for example, exponentiation of BigInts must be done with the `**` operator instead of `Math.pow`.

The nullish coalescing operator, `??`, returns its right-hand side operand when its left-hand side is `null` or `undefined`. This contrasts with the `||` operator, which would return `"string"` for all "falsy" values, such as the ones below.

```
undefined ?? "string" // -> "string"
null ?? "string" // -> "string"
false ?? "string" // -> false
NaN ?? "string" // -> NaN
```

Optional chaining makes it possible to access the nested properties of an object without having an AND check at each level. An example is `const zipcode = person?.address?.zipcode`. If any of the properties are not present, `zipcode` will be `undefined`.

## 12th Edition – ECMAScript 2021 [edit]

The 12th edition, ECMAScript 2021, was published in June 2021.<sup>[10]</sup> This version introduces the `replaceAll` method for strings; `Promise.any`, a promise combinator that short-circuits when an input value is fulfilled; `AggregateError`, a new error type to represent multiple errors at once; logical assignment operators (`??=`, `&&=`, `||=`); `WeakRef`, for referring to a target object without preserving it from garbage collection, and `FinalizationRegistry`, to manage registration and unregistration of cleanup operations performed when target objects are garbage collected; separators for numeric literals (`1_000`); and `Array.prototype.sort` was made more precise, reducing the number of cases that result in an implementation-defined sort order.

## 13th Edition – ECMAScript 2022 [\[edit\]](#)

The 13th edition, ECMAScript 2022, was published in June 2022.<sup>[11]</sup> This version introduces top-level `await`, allowing the keyword to be used at the top level of modules; new class elements: public and private instance fields, public and private static fields, private instance methods and accessors, and private static methods and accessors; static blocks inside classes, to perform per-class evaluation initialization; the `#x in obj` syntax, to test for presence of private fields on objects; regular expression match indices via the `/d` flag, which provides start and end indices for matched substrings; the `cause` property on `Error` objects, which can be used to record a causation chain in errors; the `at` method for Strings, Arrays, and TypedArrays, which allows relative indexing; and `Object.hasOwn`, a convenient alternative to `Object.prototype.hasOwnProperty`.

## 14th Edition – ECMAScript 2023 [\[edit\]](#)

The 14th edition, ECMAScript 2023, was published in June 2023.<sup>[37]</sup> This version introduces the `toSorted`, `toReversed`, `with`, `findLast`, and `findLastIndex` methods on `Array.prototype` and `TypedArray.prototype`, as well as the `toSpliced` method on `Array.prototype`; added support for `#! shebang` comments at the beginning of files to better facilitate executable ECMAScript files; and allowed the use of most Symbols as keys in weak collections.

## 15th Edition – ECMAScript 2024 [\[edit\]](#)

The 15th edition, ECMAScript 2024, was published in June 2024.<sup>[38]</sup> This version introduces the `Object.groupBy` and `Map.groupBy` static methods, `Promise.withResolvers`, and the `/v` unicode flag for regular expressions.

The `Object.groupBy` and `Map.groupBy` methods groups an iterable using the return value of a provided callback function.

```
// sample data
const arr = [
  { year: "2024", id: 0 },
  { year: "2023", id: 1 },
  { year: "2024", id: 2 },
];

const obj = Object.groupBy(arr, (el) => el.year);
console.log(obj);
// { "2024": [{ year: "2024", id: 0 }, { year: "2024", id: 2 }], "2023": [{ year: "2023", id: 1 }] }
```

`Promise.withResolvers` provides a simple way to get a promise's resolve and reject functions directly without having to assign them in the constructor.<sup>[39]</sup>

# **Web Based Application**

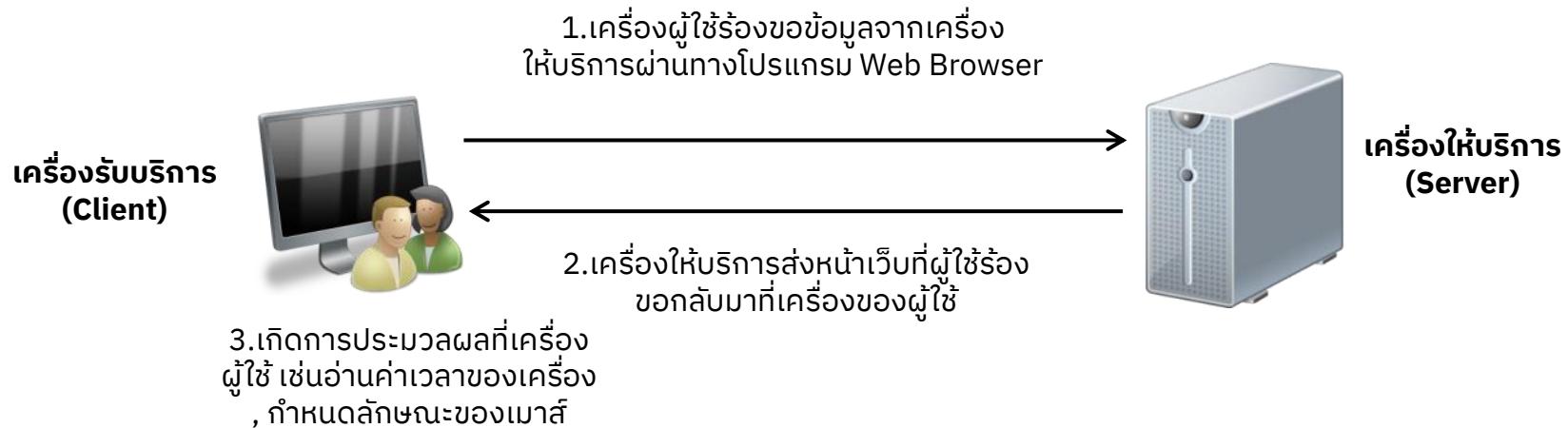
# รู้จักกับ Web Based Application

**Web-based Application** คือโปรแกรมหรือกลุ่มของโปรแกรมที่ได้รับการพัฒนาขึ้นมาเพื่อใช้งานในบริการ WWW ของระบบเครือข่ายอินเทอร์เน็ตหรือเครือข่ายอินทราเน็ต ที่ใช้โปรโตคอล TCP/IP เป็นมาตรฐานในการสื่อสารข้อมูล โดยผู้ใช้สามารถติดต่อสื่อสาร หรือเรียกใช้งานโปรแกรม Web-based Application ได้โดยใช้โปรแกรมเว็บбраузอร์ การพัฒนา Web-based Application สามารถทำได้โดยการเขียนโปรแกรมในภาษาที่ถูกออกแบบมาสำหรับการพัฒนา Application บนระบบเครือข่ายอินเทอร์เน็ต เช่น Perl , PHP, ASP , JavaScript , VB Script , JSP, JAVA และใน Application บางชนิดจะต้องมีการติดต่อกับระบบฐานข้อมูลด้วย

- ข้อมูลบนเว็บสามารถเข้าถึงได้จากผู้ชมจำนวนมากโดยไม่มีข้อจำกัดเรื่องชนิดของระบบคอมพิวเตอร์
- การนำเสนอข้อมูลบนเว็บเป็นการสื่อสารโดยตรงจากผู้ส่งสารไปยังผู้รับสารโดยใช้เวลาสั้น
- รูปแบบการนำเสนอข้อมูลมีลักษณะเป็นแบบ Hypertext และ Hypermedia ทำให้สามารถนำเสนอข้อมูลที่น่าสนใจในรูปแบบมัลติมีเดีย ที่สามารถเชื่อมโยงไปยังข้อมูลอื่น ๆ ที่เกี่ยวข้องได้
- แนวโน้มของการนำเสนอข้อมูลบนอินเทอร์เน็ตมีลักษณะ Interactive คือมีกิจกรรมที่ทำให้ผู้เข้าชมมีส่วนร่วมกับเว็บไซต์มากขึ้น เช่น Guestbook, Message board , forums etc..
- แนวโน้มของการนำเสนอข้อมูลบนอินเทอร์เน็ตมีลักษณะ Dynamic คือมีการปรับปรุงข้อมูลให้กันสมัยอยู่เสมอโดยอัตโนมัติ

การเขียนโปรแกรมหรือการประมวลผลบนเว็บ มี 2 ประเภทคือ

- 1. Client-Side Programming**
- 2. Server-Side Programming**

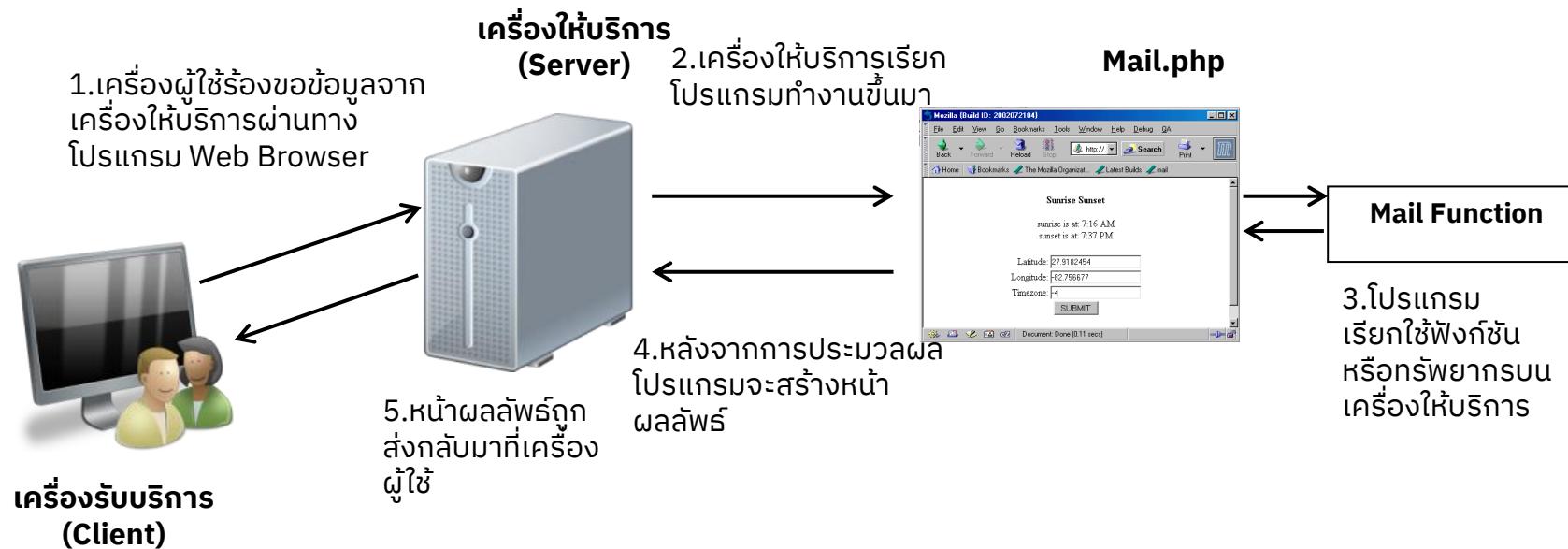


## 1. Client-Side Programming

**Client-Side Programming** คือ ลักษณะของการเขียนโปรแกรมที่จะเกิดการประมวลผลที่เครื่องรับบริการ (Client) เช่น การเขียนโปรแกรมด้วย JavaScript, VB Script

การเขียนโปรแกรมหรือการประมวลผลบนเว็บ มี 2 ประเภทคือ

- 1. Client-Side Programming**
- 2. Server-Side Programming**



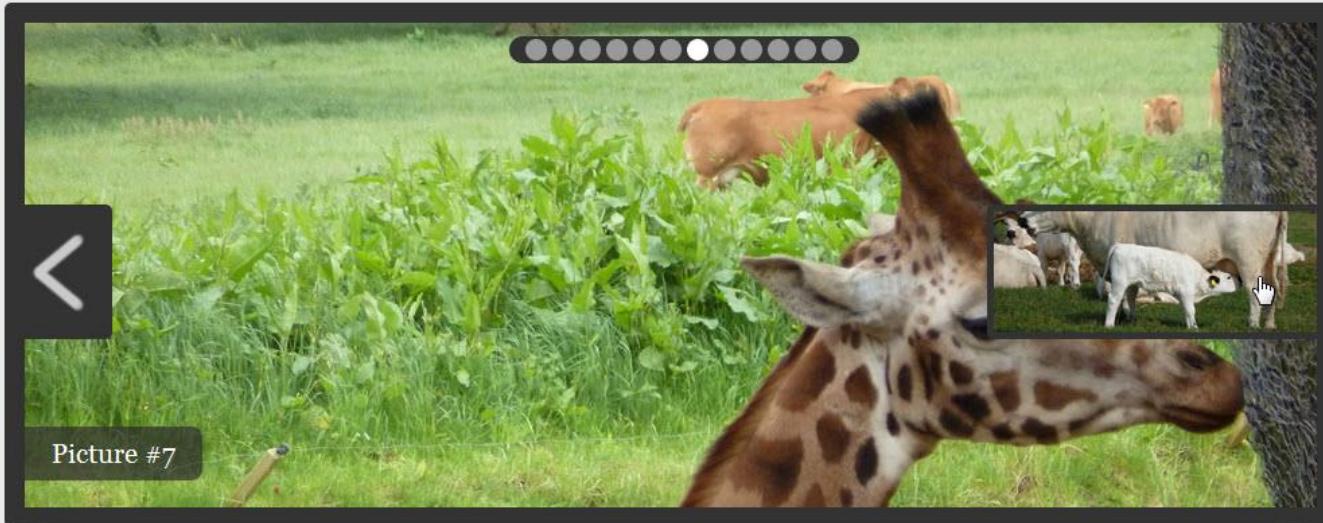
## 2. Server-Side Programming

**Server-Side Programming** คือลักษณะของการเขียนโปรแกรมที่จะเกิดการประมวลผลที่เครื่องให้บริการ (Server) เช่น การเขียนโปรแกรมด้วย PHP, Perl, ASP, JSP เป็นต้น

# Introduction to JavaScript

# CSS play - GO Slide v1.1

For IE9 (no animation), Firefox, Chrome, Safari, Opera and Safari Mobile



## Controls

Hover the image for mouse controls

Keyboard controls : left/right arrow keys (for browsers that support this) and accesskeys [A] to [L].

## Information

IE9 will not see the animation, but hopefully IE10 will support transitions.

The Safari Mobile browser cannot handle the mouse hovers or

<https://developer.mozilla.org/en-US/demos/detail/cssplay-go-slide-v11/launch>

**Javascript** คือ ภาษาจาวาชุดเล็ก ที่มีการใช้งานที่ง่ายและรวดเร็ว สามารถเขียนในไฟล์เดียวกับ โค๊ด HTML ได้ เป็นตัวช่วยให้หน้าเว็บมีความสามารถด้านต่าง ๆ ได้มากขึ้น

# What is JavaScript

**Javascript** เป็นภาษาประมวลผล (Programming language) ต่างจาก HTML และ CSS ที่เป็นภาษาสำหรับการแสดงผล (Markup language) ภาษาประมวลผลจะสามารถคำนวณ มีตัวแปร หาค่า บวก ลบ คูณ หารได้ ซึ่งสิ่งเหล่านี้จะไม่มีใน HTML จึงต้องมี JavaScript เพิ่มเข้ามา

- Javascript จะเป็นโปรแกรมย่อยเล็ก ๆ ที่สามารถแทรกอยู่ในโค๊ด HTML ได้
- JavaScript ไม่ใช่ Java และก็ส่วนภาษาไม่ได้มาจากผู้ผลิตรายเดียวกัน
- Java ผลิตจากบริษัท SUN Microsystems ตั้งแต่ปี 2534 เป็นภาษาที่ต้องมีตัวแปลงภาษา (Compiler)
- นอกจานี้ Java ยังสามารถสร้างแอพพลิเคชันทำงานบนเว็บได้ เช่นเดียวกับภาษา C หรือ VB แต่ JavaScript ทำได้แค่เป็นสคริปต์ฝังในหน้าเว็บหนึ่ง ๆ เท่านั้น
- JavaScript ผลิตมาจากการของ Netscape Communications ตั้งแต่ปี 2538
- JavaScript เป็นภาษาประเภทแปลไป ทำงานไป เมื่อൺกับการใช้ล่าม (Interpreter)
- Java จะมีขีดความสามารถสูงกว่า JavaScript แต่ JavaScript จะใช้งานบนหน้าเว็บได้ง่ายกว่า เพราะเพียงแค่เพิ่มโค๊ด JavaScript ลงไปในหน้าเว็บก็สามารถเรียกดูผลลัพธ์ได้ ต่างจาก Java ที่ต้องใช้ Compiler แปลเป็นภาษาเครื่องก่อน

# ຕាំងនៃការបើយោ JavaScript

ໃນគូរសរោះនៃ HTML ទាសាមារណ៍ដំឡើង JavaScript ទៅក្នុងក្រុងក្រាល។

## 1. តាំងនៃស៊ីនុវត្ថុនៃការបើយោ HTML

```
<head>
    <script language="JavaScript">
        .....ការសំណើនៅ Javascript .....
    </script>
</head>
```

Ex.

```
<head>
    <script language="JavaScript">
        alert("Hello JavaScript");
    </script>
</head>
```

# คำแนะนำในการเขียน JavaScript

ในโครงสร้างของ HTML เราสามารถผัง JavaScript ไว้ที่คำแนะนำได้โดยได้ดังนี้

## 2. คำแนะนำส่วนตัวของเอกสาร HTML

```
<body>
    <script language="JavaScript">
        .....คำสั่ง Javascript .....
    </script>
</body>
```

Ex.

```
< body>
    <script language="JavaScript">
        alert("Hello JavaScript");
    </script>
</ body>
```

# ตำแหน่งการเขียน JavaScript

ในโครงสร้างของ HTML เราสามารถฝัง JavaScript ไว้ที่ตำแหน่งใดก็ได้ ดังนี้

## 3. เป็นค่าในอีเวนต์ (Event) ของแท็กคำสั่ง HTML

```
<body onload="คำสั่ง JavaScript">  
  
</body>
```

## 4. เก็บไว้ในไฟล์แยกต่างหาก โดยเก็บไว้ในนามสกุล .js

```
<script language="JavaScript" type="text/javascript" src="java1.js"></script>
```

# วิธีการเขียน JavaScript

คำสั่ง JavaScript จะเป็นคำสั่งที่กำหนดช้ายไปขวา บันลุณล่าง เรียงลำดับลงไปเรื่อย ๆ แต่ละคำสั่งจะจบด้วยเครื่องหมาย Semicolon (;) เช่น

```
<script language="JavaScript">
    document.write("Hello JavaScript"); alert("Hello Johny");
</script>
```

ถ้าเขียนเว็บเบราว์เซอร์ก็ต้องมี Semicolon ก็ได้

```
<script language="JavaScript">
    document.write("Hello JavaScript")
    alert("Hello Johny")
</script>
```

# กลุ่มคำสั่ง JavaScript Blocks

คำสั่ง JavaScript สามารถจัดกลุ่มได้ เพื่อมองทั้งกลุ่มเป็นคำสั่งเดียวโดยใช้เครื่องหมาย Curly bracket{...}

```
<script language="JavaScript">
{
    document.write("Hello JavaScript1")
    document.write("Hello JavaScript2")
    document.write("Hello JavaScript3")
}
</script>
```

# ตัวพิมพ์เล็ก-ใหญ่ (JavaScript in Case-sensitive)

ภาษา JavaScript เรื่องตัวพิมพ์เล็ก-ใหญ่ ถือว่าสำคัญ เพราะเป็นภาษาแบบ Case-sensitive เช่น Myworld กับ MyWorld จะถือว่าเป็นคนละตัวกัน

การตั้งชื่อตัวแปรต่าง ๆ นิยมใช้เป็นตัวพิมพ์เล็ก เพื่อตัดปัญหาการเรียกชื่อผิด

```
<script language="JavaScript">
  var name = "John Carter"
  var age = 28
</script>
```

# The core **JavaScript** language

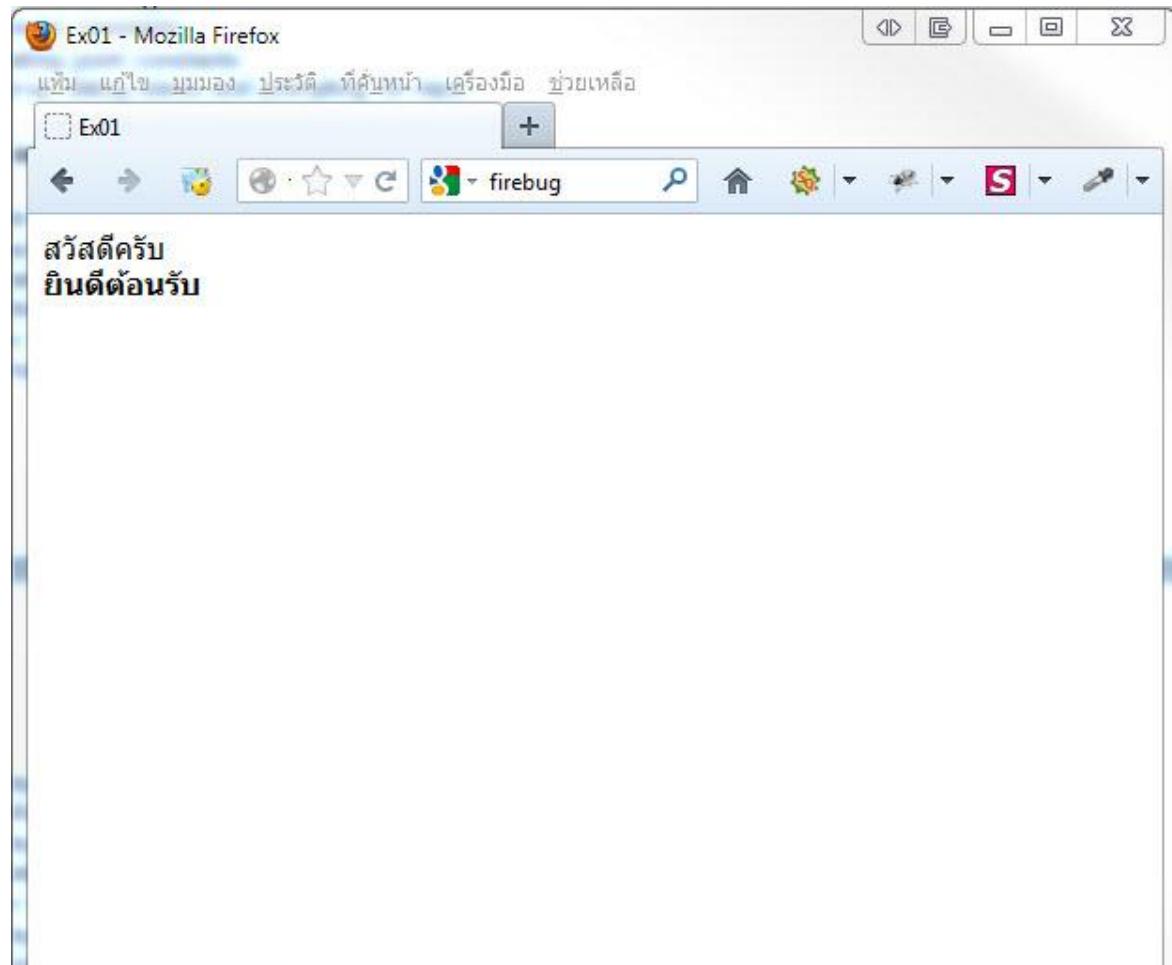
# คำสั่งการแสดงผลอักหน้าจอ

คำสั่งที่ใช้ในการแสดงผลข้อความบนหน้าจอคือ

```
document.write("ข้อความ");
```

Example

```
document.write("สวัสดีครับ");
document.write("<b>ยินดีต้อนรับ</b><br>");
```



EditPlus - [C:\Users\Samit\Desktop\Javascript\ExampleCode\ex01.html]

```
File Edit View Document Project Tools Browser Zen Coding Window Help
B I U A nb Hx W TAB CD ✓
Directory Cliptext
[C:]
C:\ AppServ www 21millionaire201 meeting include
config.inc.php function.inc.php jquery.js jscript-admin.js style.css
All Files (*.*)
ex01.html ex01.html
For Help, press F1 9 col 31 14 22 PC UTF-8
```

1 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
2 <html>
3 <head>
4 <title> Ex01 </title>
5 </head>
6
7 <body>
8 <script language="javascript">
9 document.write("สวัสดีครับ<br>");
10 document.write("<b>ยินดีต้อนรับ</b><br>");
11 </script>
12 </body>
13 </html>

# Character set

การเข้ารหัสสำหรับแสดงผลภาษาต่าง ๆ ของ JavaScript สามารถทำได้ดังนี้

```
<script type="text/javascript" src="[path]/myscript.js" charset="utf-8"></script>
```

## Declaring variable (การประกาศตัวแปร)

**var** ชื่อตัวแปร;

Example

```
var name = "samit koyom";  
var age = 28;
```

# Reserved words (คำสংবৃ)

abstract (\*)  
as (2)  
boolean  
break  
byte  
case  
catch  
char  
class (2)  
continue  
const (2)  
debugger (\*)  
default  
delete  
do  
double  
else  
enum (\*)  
export (2)  
extends (2)  
false

final  
finally  
float  
for  
function  
goto (\*)  
if  
implements (\*)  
import (2)  
in  
instanceof  
int  
interface (2)  
is (2)  
long  
namespace (2)  
native (\*)  
new  
null  
package (2)  
private (2)

protected (\*)  
public (2)  
return  
short  
static (2)  
super (2)  
switch  
synchronized (\*)  
this  
throw  
throws (\*)  
transient (\*)  
true  
try  
typeof  
use (2)  
var  
void  
volatile (\*)  
while  
with

# คำสั่งแสดงหน้าต่าง Popup (alert)

```
alert("ข้อความ");
```

## Example

```
alert("สวัสดีครับทุกคน");
```

# คำสั่งแสดงผลทาง Console

```
console.log("ข้อความ");
```

## Example

```
console.log("สวัสดีครับทุกคน");
```

# เมธอด confirm()

เป็นกล่องแสดงข้อความเพื่อให้ผู้ใช้เลือก และคืนค่าถูก (true), ผิด (false) กลับมาให้โปรแกรมใช้งานต่อ มีรูปแบบการใช้งานดังนี้

confirm("ข้อความ");

## Example

confirm("Are you OK ?");

```
8 <body>
9 <h1>Hello All</h1>
10
11 <script language="javascript">
12 var qt = confirm("Are you OK ?");
13 if(qt==true){
14     document.writeln("Wellcom to my site<br>");
15 }else{
16     document.writeln("Quit from my site");
17 }
18 </script>
19
20 </body>
```

# **Pop-up dialog boxes and prompting for input**

## **เมธอด prompt()**

เป็นกล่องแสดงข้อความเพื่อให้ผู้ใช้กรอกข้อมูลมาให้โปรแกรมใช้งานต่อ มีรูปแบบการใช้งานดังนี้

```
prompt("ข้อความ", "ค่าเริ่มต้น");
```

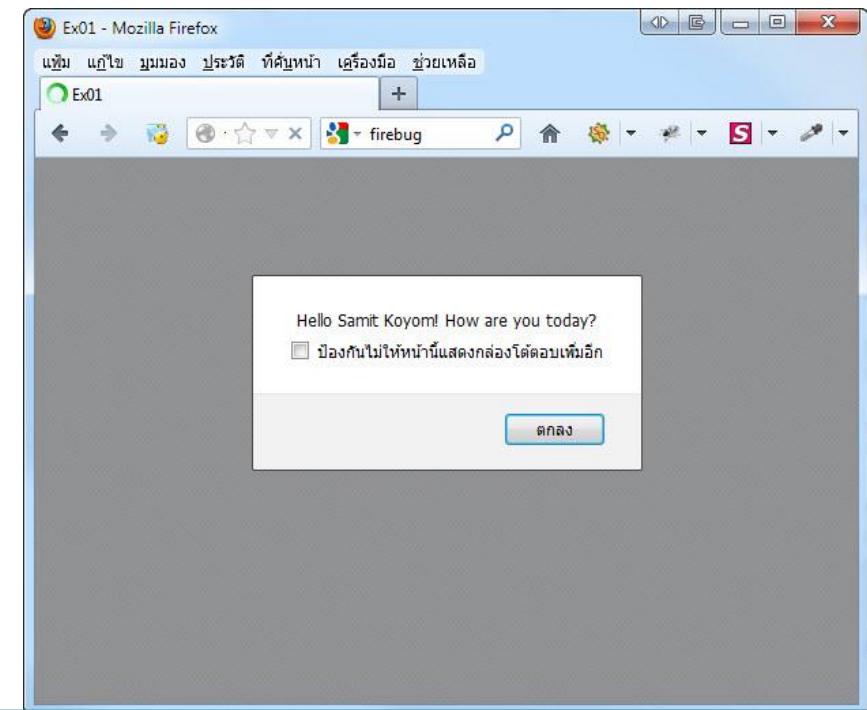
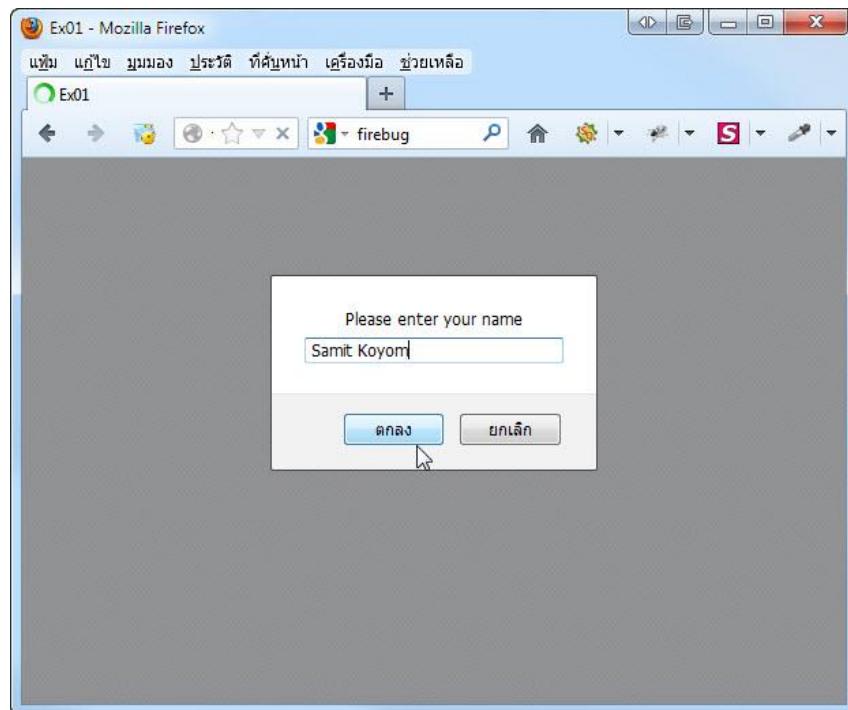
### **Example**

```
n=window.prompt("Please enter your name", "Samit Koyom");
alert("Hello " + n + "! How are you today?");
```

# Pop-up dialog boxes and prompting for input

## Example

```
n=window.prompt("Please enter your name", "Samit Koyom");
alert("Hello " + n + "! How are you today?");
```



# Arithmetic and Operator

# Arithmetic Operators

**y = 5;**

Operator	Description	Example	Result of x	Result of y
+	Addition	x=y+2	7	5
-	Subtraction	x=y-2	3	5
*	Multiplication	x=y*2	10	5
/	Division	x=y/2	2.5	5
%	Modulus (division remainder)	x=y%2	1	5
++	Increment	x=++y	6	6
		x=y++	5	6
--	Decrement	x=--y	4	4
		x=y--	5	4

# Assignment Operators

**x = 10 , y = 5;**

Operator	Example	Same As	Result
=	x=y		x=5
+=	x+=y	x=x+y	x=15
-=	x-=y	x=x-y	x=5
*=	x*=y	x=x*y	x=50
/=	x/=y	x=x/y	x=2
%=	x%=y	x=x%y	x=0

# Comparison Operators    $x = 5;$

Operator	Description	Comparing	Returns
==	is equal to	$x == 8$	<i>false</i>
		$x == 5$	<i>true</i>
=====	is exactly equal to (value and type)	$x === "5"$	<i>false</i>
		$x === 5$	<i>true</i>
!=	is not equal	$x != 8$	<i>true</i>
!==	is not equal (neither value or type)	$x !== "5"$	<i>true</i>
		$x !== 5$	<i>false</i>
>	is greater than	$x > 8$	<i>false</i>
<	is less than	$x < 8$	<i>true</i>
>=	is greater than or equal to	$x >= 8$	<i>false</i>
<=	is less than or equal to	$x <= 8$	<i>true</i>

# Logical Operators

x = 6 and y = 3

Operator	Description	Example
&&	and	(x < 10 && y > 1) is true
	or	(x==5    y==5) is false
!	not	!(x==y) is true

## Conditional Operator

*variablename=(condition)?value1:value2*

example

`voteable=(age<18)?"Too young":"Old enough"`

# JavaScript flow control

# Conditional

In JavaScript we have the following conditional statements:

- **if statement** - use this statement to execute some code only if a specified condition is true
- **if...else statement** - use this statement to execute some code if the condition is true and another code if the condition is false
- **if...else if....else statement** - use this statement to select one of many blocks of code to be executed
- **switch statement** - use this statement to select one of many blocks of code to be executed

# If Statement

## Syntax

```
if (condition)
{
    code to be executed if condition is true
}
```

## Example

```
if (time<20)
{
    x="Good day";
}
```

# If...else Statement

## Syntax

```
if (condition)
{
    code to be executed if condition is true
}
else
{
    code to be executed if condition is not true
}
```

# If...else Statement

## Example

```
if (time<20)
{
    x="Good day";
}
else
{
    x="Good evening";
}
```

# If...else if...else Statement

## Syntax

```
if (condition1)
{
    code to be executed if condition1 is true
}
else if (condition2)
{
    code to be executed if condition2 is true
}
else
{
    code to be executed if neither condition1 nor condition2 is true
}
```

# If...else if...else Statement

## Example

```
if (time<10)
{
    x="Good morning";
}
else if (time<20)
{
    x="Good day";
}
else
{
    x="Good evening";
}
```

# Switch Statement

## Syntax

```
switch(n)
{
    case 1:
        execute code block 1
        break;
    case 2:
        execute code block 2
        break;
    default:
        code to be executed if n is different from case 1 and 2
}
```

# Switch Statement

## Example

```
var day=new Date().getDay();
switch (day)
{
    case 0:
        x="Today it's Sunday";
        break;
    case 1:
        x="Today it's Monday";
        break;
    case 2:
        x="Today it's Tuesday";
        break;
```

```
case 3:
    x="Today it's Wednesday";
    break;
case 4:
    x="Today it's Thursday";
    break;
case 5:
    x="Today it's Friday";
    break;
case 6:
    x="Today it's Saturday";
    break;
}
```

# The default Keyword

## Example

```
var day=new Date().getDay();
switch (day)
{
case 6:
    x="Today it's Saturday";
    break;
case 0:
    x="Today it's Sunday";
    break;
default:
    x="Looking forward to the Weekend";
}
```

# การกำช้อด (for) ด้วย javascript

## Syntax

```
for();
```

## Example

```
var a;
for(a=1;a<=10;a++){
document.write("Welcome<br>");
}
```

# การกำช้ำ (while) ด้วย javascript

## Syntax

```
while (variable<endvalue)
{
  code to be executed
}
```

## Example

```
var i =0;
while (i<5)
{
  document.write(i+ “Welcome<br>”);
  i++;
}
```

# JavaScript functions

# การสร้างฟังก์ชันและเรียกใช้งานใน javascript

`function name()`

## Example

```
function wakeup() {  
    alert("Hello");  
}
```

## Use in HTML

```
<input type="button" value="Click Now" onclick="wakeup()" />
```

# ตัวอย่าง

```
5 <title>Basic JavaScript 1</title>
6
7 <script language="javascript">
8 var number = 10;
9
10 function wakeup(){
11     //alert("Hello"); (1)
12     document.myform.result.value="Hello " + number; // document เป็นการอ้างถึงเอกสารบนหน้าเพจ
13     number++;
14 }
15
16 </script>
17
18 </head>
19
20 <body>
21 <h1>Hello All</h1>
22
23 <form name="myform">
24     <input type="text" name="result"/>
25     <input type="button" value="Click Now" onclick="wakeup()" />
26 </form>
27
28 </body>
```

# Array in JavaScript

# Basic Array

```
var name = ["a", "b", "c", "d"]
```

```
var name = new Array("a", "b", "c", "d")
```

```
console.log(name[0])
```

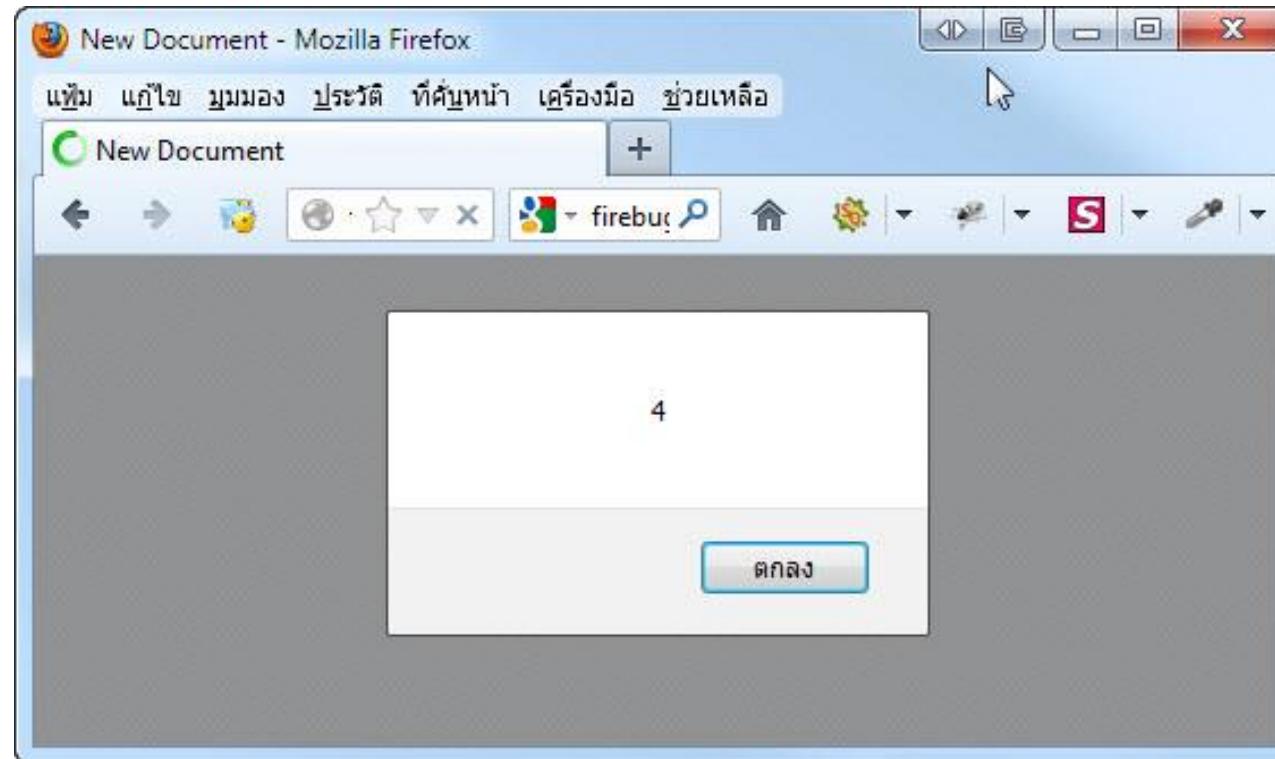
## Example

```
var firstname = ["John", "Jane", "Joe", "Joom"];
var i;
for(i=0; name[i]; i++){
    document.write("Hello "+name[i]+ "<br>");
}
```

# Array length (การหาขนาด array)

## Example

```
var name = ["John", "Jane", "Joe", "Joom"];
alert(name.length);
```



# Object in JavaScript

# Basic Object in JavaScript

```
var name = { key1: "value1", key2: "value2" }
```

# การเข้าถึง tag div ,span

```
document.getElementById("ชื่อ id ของ div หรือ span");
```

## Example

```
document.getElementTagName("ชื่อ name ของ div หรือ span");
```



```
document.getElementById("resultarea").innerHTML="Hello";
```

```
<div id="resultarea">แสดงผลที่นี่</div>
```

# การรับค่า input จากฟอร์ม

## Syntax

```
document.formname.inputname.value;
```

```
document.myform.result.value;
```

```
<form name="myform">  
Result<input type="text" name="result" />  
</form>
```

# ตัวอย่างการรับค่า input จากฟอร์ม

```
5 <title>Basic JavaScript 1</title>
6
7 <script language="javascript">
8
9 function wakeup(){
10    var name= document.myform.name.value;
11    document.myform.result.value="Hello "+name;
12
13    document.myform.name.value = "";
14 }
15 </script>
16
17 </head>
18
19
20 <body>
21 <h1>Hello All</h1>
22
23 <form name="myform">
24 Name <input type="text" name="name"/>
25 Result <input type="text" name="result" />
26 <input type="button" value="Click Now" onclick="wakeup()" />
27 </form>
28
```

ชื่อ-สกุล

สามีตร ໂກຍມ

ที่อยู่

2 ลาดพร้าว 65/1  
วังทองหลาง กทม.

บันทึก

# การเข้าถึง tag div ,span

```
6
7 <script language="javascript">
8
9 function wakeup(){
10    var name= document.myform.name.value;
11    document.getElementById("resultarea").innerHTML="Hello "+name;
12    document.myform.name.value = "";
13 }
14 </script>
15
16 </head>
17
18 <body>
19 <h1>Hello All</h1>
20
21 <form name="myform">
22 Name <input type="text" name="name"/>
23 <input type="button" value="Click Now" onclick="wakeup()" />
24 </form>
25
26 <hr />
27 <div id="resultarea">แสดงผลที่นี่</div>
28
29
```

# JavaScript Events

# Events

By using JavaScript, we have the ability to create dynamic web pages. Events are actions that can be detected by JavaScript.

Every element on a web page has certain events which can trigger a JavaScript. For example, we can use the onclick event of a button element to indicate that a function will run when a user clicks on the button.

## **Examples of events:**

- Clicking a button (or any other HTML element)
- A page is finished loading
- An image is finished loading
- Moving the mouse-cursor over an element
- Entering an input field
- Submitting a form

# Events

## Mouse Events

Event	Attribute	Description
click	<a href="#"><u>onclick</u></a>	The event occurs when the user clicks on an element
dblclick	<a href="#"><u>ondblclick</u></a>	The event occurs when the user double-clicks on an element
mousedown	<a href="#"><u>onmousedown</u></a>	The event occurs when a user presses a mouse button over an element
mousemove	<a href="#"><u>onmousemove</u></a>	The event occurs when a user moves the mouse pointer over an element
mouseover	<a href="#"><u>onmouseover</u></a>	The event occurs when a user mouse over an element
mouseout	<a href="#"><u>onmouseout</u></a>	The event occurs when a user moves the mouse pointer out of an element
mouseup	<a href="#"><u>onmouseup</u></a>	The event occurs when a user releases a mouse button over an element

# Events

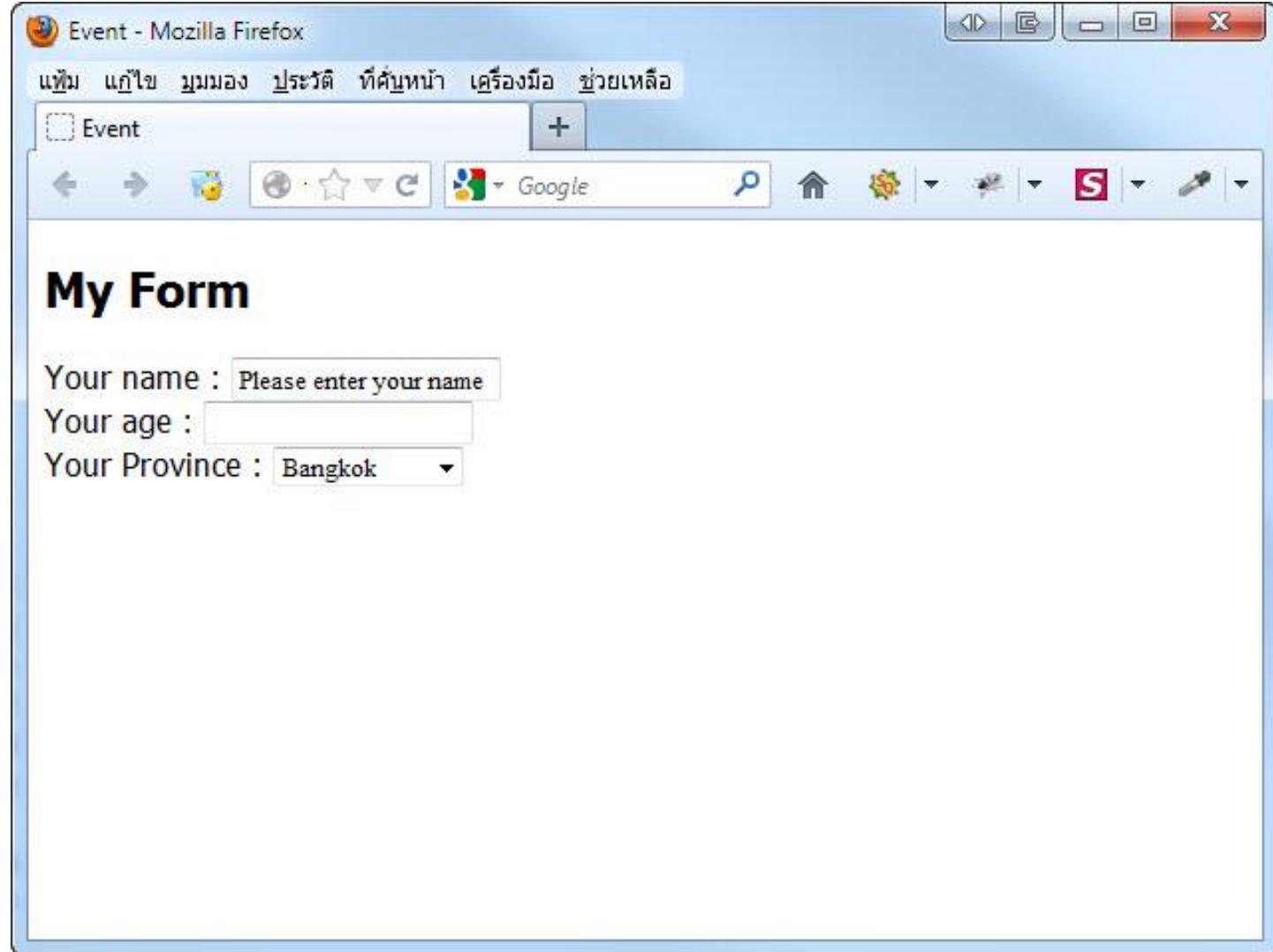
## Keyboard Events

Event	Attribute	Description
keydown	<a href="#"><u>onkeydown</u></a>	The event occurs when the user is pressing a key or holding down a key
keypress	<a href="#"><u>onkeypress</u></a>	The event occurs when the user is pressing a key or holding down a key
keyup	<a href="#"><u>onkeyup</u></a>	The event occurs when a keyboard key is released

# Form Events

Event	Attribute	Description
blur	<a href="#"><u>onblur</u></a>	The event occurs when a form element loses focus
change	<a href="#"><u>onchange</u></a>	The event occurs when the content of a form element, the selection, or the checked state have changed (for <input>, <select>, and <textarea>)
focus	<a href="#"><u>onfocus</u></a>	The event occurs when an element gets focus (for <label>, <input>, <select>, <textarea>, and <button>)
reset	onreset	The event occurs when a form is reset
select	<a href="#"><u>onselect</u></a>	The event occurs when a user selects some text (for <input> and <textarea>)
submit	onsubmit	The event occurs when a form is submitted

# Events



# Example

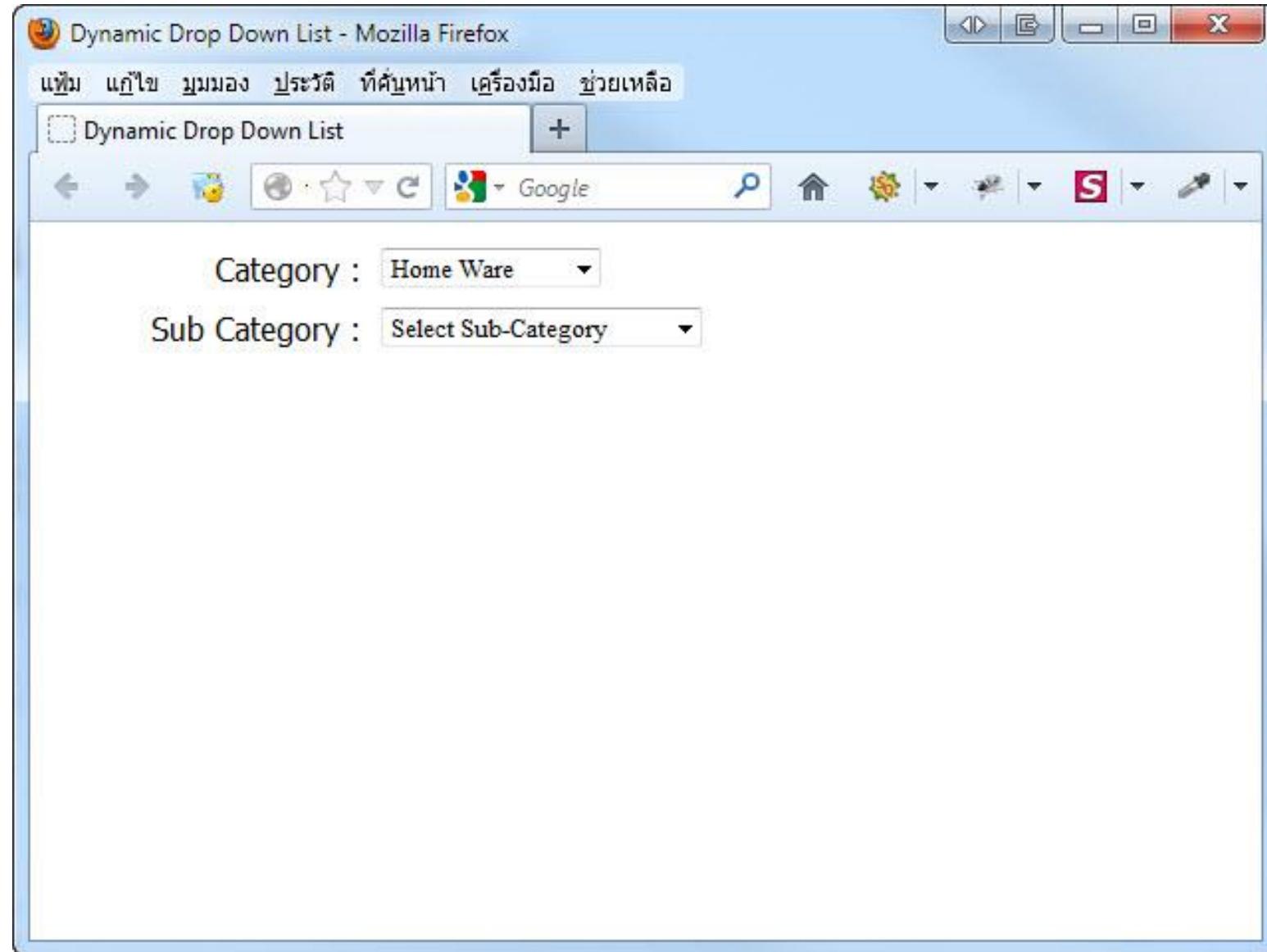
```
<script type="text/javascript">
    function blurform(){
        var nameval = document.myform.name.value;
        if(nameval == ""){
            document.myform.name.value = "Please enter your name";
        }
    }

    function focusform(){
        if(nameval == ""){
            document.myform.name.value = "";
        }
    }
</script>
```

# Events

```
<body onload="blurform()">
<h2>My Form</h2>
<form name="myform" id="myform" action="#" method="post">
Your name : <input type="text" name="name" id="name" onblur="blurform()"
onfocus="focusform()"/><br />
Your age : <input type="text" name="age" id="age"/><br />
Your Province :
<select name="select" id="select">
<option value="1">Bangkok</option>
<option value="2">Nontaburi</option>
<option value="3">Samutprakan</option>
<option value="4">Rayong</option>
</select>
</form>
<body>
```

# Events





# AGENDA

Classes

Arrow functions

String Templates

Promises

Modules



Destructuring

Consts

Let

Parameters

Transpilation

# 1. No semicolon

# semicolon ກາຍໄປ

```
export default function applyMiddleware(...middlewares) {
  return (createStore) => (reducer, initialState, enhancer) => {
    var store = createStore(reducer, initialState, enhancer)
    var dispatch = store.dispatch
    var chain = []

    var middlewareAPI = {
      getState: store.getState,
      dispatch: (action) => dispatch(action)
    }
    chain = middlewares.map(middleware => middleware(middlewareAPI))
    dispatch = compose(...chain)(store.dispatch)

    return {
      ...store,
      dispatch
    }
  }
}
```

## 2. การสร้างตัวแปรแบบใหม่

# var, let ,const

# ปัญหาของ Var

เมื่อlob เขตการทำงานเป็นแบบ Global Scope

```
var name = "John";
{
  var name = "Jany";
}

console.log(name); // Jany
```

```
var name = "John";
function a(){
  var name = "Jany";
}

console.log(name); // John
```

```
var messages = ['Hello', 'JavaScript', 'es2015'];
for (var i = 0; i < messages.length; i++) {
    console.log(i)
    setTimeout(function() {
        console.log(i)
        console.log(messages[i])
    }, 1000);
}

// undefined
// undefined
// undefined
```

# let & const

## let

- *let* มี slogan ว่า **let is the new var**
- *let* เป็น block-scope นั่นหมายความว่า scope ของมันจะอยู่แค่ภายในบล็อกๆ หนึ่ง เท่านั้น ไม่ใช้ภายใน function เหมือนกับ *var*
- *let* หากเรียกใช้ต่อน้ำที่ยังไม่ได้ประกาศค่า จะขึ้น Error "ไม่เหมือนกับ *var* ที่จะขึ้น *undefined*"
- *let* ไม่มีปัญหาการแพร์ตัวแปรใน loop ในขณะที่ *var* มีปัญหานี้ เช่น

# ຕົວຢ່າງການໃຊ້ let

ເນື້ອບເຂດການກຳຈານເປັນແບບ block scope

```
let name = "John";
{
  let name = "Jany";
  console.log(name)
}
console.log(name)
// jany
// john
```

```
let messages = ['Hello', 'JavaScript', 'es2015'];
for (let i = 0; i < messages.length; i++) {
  //console.log(i)
  setTimeout(function() {
    //console.log(i)
    console.log(messages[i])
  }, 1000);
}
// Hello
// JavaScript
// es2015
```

# **const**

- *const* หากจะใช้ ต้องกำหนดค่าให้มันด้วยทุกครั้ง ไม่สามารถที่จะตั้ง variable เฉยๆได้
- *const* กำหนดค่าได้แค่ครั้งเดียว นั่นก็คือตอนเริ่มต้นตั้งชื่อตัวแปร จะนั้น *const* หมายความว่าใช้เก็บตัวแปรที่เป็นค่าคงที่ไม่มีการเปลี่ยนแปลง

# ตัวอย่างการใช้ const

มีขอบเขตการทำงานเป็นแบบ block scope และกำหนดค่าแล้วเปลี่ยนแปลงไม่ได้

```
// การใช้งาน const  
const firstname  
console.log(firstname) // SyntaxError: Missing initializer
```

```
// การใช้งาน const  
const firstname = "Samit"  
firstname = "Somchai"  
// TypeError: Assignment to constant variable.
```

```
// การใช้งาน const  
const person = {}  
person.name = "Somkid"  
person.email = "somkid@email.com"  
person.tel = "089389082902"  
console.log(person)
```

# 3 .Arrow Functions in ES6

# Arrow Functions in ES6

Arrow function (=>) หรืออาจเรียกว่า fat arrow function เป็น short hand function โดยใช้เครื่องหมาย =>

ตัวอย่างการสร้างฟังก์ชันใน ES5

```
var greet = function(name, message) {  
    return message + name  
}
```

เขียนด้วย Arrow function ใน ES6

```
var greet = (name, message) => {  
    return message + name  
}
```

ถ้าฟังก์ชันมีการ return เพียงอย่างเดียว

```
var greet = (name, message) => message + name
```

ถ้าฟังก์ชันมีการ return เพียงอย่างเดียว และมี parameter เดียว

```
var greet = message => message
```

Example

```
var square = x => x * x
```

# 4. Default Parameters

# Default Parameters in ES6

```
function sayHi(name="Chai") {  
    console.log('Hello ' + this.name);  
}
```

ชิ้นสถาเป็นแบบเก่า เราก็ต้องมาเช็คว่าค่า name มีค่าหรือไม่ แบบนี้

```
function sayHi(name) {  
    if(name === undefined) {  
        this.name = 'Chai'  
    }  
    console.log('Hello ' + this.name);  
}
```

# Default Parameters in ES6

```
var link = function (height, color, url) {  
    var height = height || 50  
    var color = color || 'red'  
    var url = url || 'http://azat.co'  
    ...  
}
```

```
var link = function(height = 50, color = 'red', url = 'http://azat.co') {  
    ...  
}
```

# 5. Template Literals in ES6

# Template Literals in ES6

## Multi Line String

การใช้งาน String แบบหลายบรรทัด โดยปกติถ้าเป็น ES5 เวลาเราจะกำหนด String  
หลายบรรทัด เราจะกำหนดด้วย

```
var htmlContent =  
  '<!DOCTYPE html>\n' +  
  '<html>\n' +  
  '<head>\n' +  
  '  <meta charset="UTF-8">\n' +  
  '  <title></title>\n' +  
  '</head>\n' +  
  '<body>\n' +  
  '</body>\n' +  
  '</html>\n';
```

# Template Literals in ES6

แต่ถ้าเป็น ES6 เราสามารถกำหนด โดยใช้เครื่องหมาย `` แบบนี้ได้เลย

```
const htmlContent = `<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title></title>
</head>
<body>
</body>
</html>`;
```

# Interpolation String

## Interpolation

โดยปกติเวลาเราจะต้องเติม String ใน ES5 เราจะทำแบบนี้

```
function printInformation(name, position, company) {  
    console.log(name + ' work as ' + position + ' at ' + company);  
}
```

แต่ถ้าเป็น ES6 เราสามารถใช้ความสามารถของ String Template แทรกตัวแปรลงไปใน String ได้เลย คล้ายๆในหลายภาษา เช่น Ruby ได้แบบนี้

```
function printInformation(name, position, company) {  
    console.log(` ${name} work as ${position} at ${company}`);  
}
```

# 6. Destructuring Assignment in ES6

เป็น Pattern ที่ใช้ในการกำหนดค่าให้กับตัวแปร โดยจะแบ่งค่ากับ value หรือ properties

# Destructuring Assignment in ES6

ตัวอย่างใน ES 5

```
var scores = [55,65,82];
var s1 = scores[0];
var s2 = scores[1];
console.log(s1, s2);
```

ตัวอย่างใน ES 6

```
const scores = [55, 65, 82]
const [s1, s2] = scores
console.log(s1, s2);
```

# Destructuring Assignment in ES6

ชื่นถ้าหากเปรียบเทียบกับการใช้ ES5 แบบเดิม จะต้องเขียนแบบนี้

```
var player = {  
    name: "Lionel Messi",  
    club: "Barcelona"  
};  
  
var name = player.name;  
var club = player.club;  
  
console.log(name);      // "Lionel Messi"  
console.log(club);     // "Barcelona"
```

# Destructuring Assignment in ES6

## Destructoring

เป็น Pattern ที่ใช้ในการกำหนดค่าให้กับตัวแปร โดยจะแมปค่ากับ value หรือ properties

ตัวอย่างเช่น

```
let player = {  
    name: "Lionel Messi",  
    club: "Barcelona"  
};  
  
let { name, club } = player;  
  
console.log(name);      // "Lionel Messi"  
console.log(club);     // "Barcelona"
```

# Destructuring Assignment in ES6

```
// Return object
function createuser(){
    return {name: 'Samit', age: 30}
}

const {name, age } = createuser()
console.log(name, age)
```

# 7. Rest and Spread Operator

# Rest and Spread Operator in ES6

```
// Spread  
const arr = [4, 5, 6]  
const append = [1, 2, 3, arr] // cannot do this  
console.log(append)
```

```
// Spread  
const arr = [4, 5, 6]  
const append = [1, 2, 3, ...arr] // spread operator  
console.log(append)
```

# Rest and Spread Operator in ES6

```
// Rest Parameter
function howManyArgs(...args){
    console.log(args.length);
    console.log(args)
}
```

```
howManyArgs()
howManyArgs(4)
howManyArgs(3, 5)
howManyArgs(3, 5, 8, 2, 7)
```

# Rest and Spread Operator in ES6



```
1 // Rest Parameter  
2 function multiply(multiplier, ...array){  
3     return array.map(e => multiplier * e)  
4 }  
5  
6 const result = multiply(2, 10, 20 ,30)  
7 console.log(result)
```

# 8. Classes in ES6

# Classes in ES6

ปกติใน ES5 เราจะไม่มีคลาส ทำให้ JavaScript ยังไม่เป็น OOP เดี๋นตัว ใช้ความสามารถของ prototype แทน แต่ถ้าเป็น ES6 เราสามารถสร้าง class ได้แล้ว มีทั้ง constructor และ extends เมื่อเทียบภาษาอื่นๆอย่าง Java

ซึ่งเมื่อก่อนเราสร้างคลาส ด้วย function ประมาณนี้

```
function Person(name) {  
    this.name = name;  
}  
  
Person.prototype.sayHi = function() {  
    console.log('Hi, my name is ' + this.name);  
}
```

# Classes in ES6

แต่ถ้าเป็น ES6 เราสามารถใช้ Class syntax และ constructor ได้เลย แบบนี้

```
class Person {  
    constructor(name) {  
        this.name = name;  
    }  
  
    sayHi() {  
        console.log('Hi, my name is ' + this.name);  
    }  
}
```

# Classes in ES6

```
// ตัวอย่างการสร้าง class
class Person {
    constructor(name){
        this.name = name
    }

    sayHi(){
        console.log('Hi, My name is ' + this.name)
    }
}
```

```
// การสร้าง object หรือ instance
const p = new Person('Samit')
console.log(p.name)
p.sayHi()
```

# Classes in ES6

```
class baseModel {  
  constructor(options = {}, data = []) { // class constructor  
    this.name = 'Base'  
    this.url = 'http://azat.co/api'  
    this.data = data  
    this.options = options  
  }  
  
  getName() { // class method  
    console.log(`Class name: ${this.name}`)  
  }  
}
```

# 9. Modules in ES6

# Modules in ES6

JavaScript ไม่เคยจัดการโมดูลได้ด้วยตัวเองมาก่อนต้องทำผ่านไลบรารีอย่าง CommonJS หรือ AMD การมาของ ES2015 มาพร้อมกับการสนับสนุนการทำงานกับโมดูลในตัว ตอนนี้คุณสามารถใช้ ES2015 เพื่อ import/export ของจากไฟล์หนึ่งไปอีกไฟล์หนึ่งได้แล้ว ดังนี้

```
1 // dog.js
2 export const DEFAULT_COLOR = 'white'
3 export function walk() {
4   console.log('Walking...')
5 }
6
7 // main.js
8 // เลือกนำเข้าเฉพาะ DEFAULT_COLOR
9 import { DEFAULT_COLOR } from './dog.js'
10
11 // main.js
12 // นำเข้าทุกสรรพสิ่งที่ export จาก dog
13 // และตั้งชื่อใหม่ให้ว่า lib
14 import * as lib from './dog.js'
15
```

# Modules in ES6

ถ้าหากโนดูลนั้นมีแค่สิ่งเดียวที่อยาก export ทำได้ดังนี้

```
1 // circle.js
2 // สังเกตคำว่า default
3 export default class Circle {
4     area() {
5
6     }
7 }
8
9 // main.js
10 import Circle from './circle.js'
11
```

# Modules in ES6

ES2015 module นั้นคลาด มันมีการตรวจสอบว่าเรา import อะไรเข้ามาบ้าง ถ้าสิ่งไหนไม่ได้ import มันจะไม่นำมารวม ทำให้ไฟล์มีขนาดเล็กกว่าการใช้ CommonJS module เนื่องจาก CommonJS จะ import ทุกสรรพสิ่งที่ในคูลนั้น export ออกมานะ

```
1 // dog.js
2 export const DEFAULT_COLOR = 'white'
3 export function walk() {
4   console.log('Walking...')
5 }
6
7 // main.js
8 // เลือกฟังก์ชัน walk
9 import { walk } from './dog.js'
10 walk()
11
12 // ผลลัพธ์สุดท้ายจะเป็น...
13 // สังเกตว่าไม่มี DEFAULT_COLOR ติดมากับ
14 function walk()
15   console.log('Walking...')
16 }
17 walk()
```

# Export before declarations

```
export default class User { // มีการเติมคำว่า default ไปหลัง export  
  constructor(name) {  
    this.name = name;  
  }  
  
  sayHi(){  
    console.log('Hi '+ this.name)  
  }  
}
```

# Export before declarations

```
// Import Class ที่มีการกำหนด Default  
import User from './user.js'
```

```
// สร้าง object  
const obj = new User('John')  
console.log(obj.name);  
obj.sayHi()
```

# 10. Basic Arrays

# การจัดการ array

- Join
- Reduce
- Concat
- Push
- Pop
- Shift
- Unshift
- Splice
- Slice
- Mutation
- Foreach
- For of
- indexOf
- findIndexOf

# 11. Array Map

# 12. Async and Callback

# 13. Async and Promise

# 14. Async and Await

# ขั้นตอนการ build ด้วย webpack

การติดตั้ง webpack ลงในโปรเจกต์

---

step 1: npm init -y

step 2: npm install webpack webpack-cli --save-dev

step 3: เพิ่มคำสั่งนี้เข้าไปใน package.json

```
"build": "webpack --mode development --entry ./src/js/main.js --output  
./dist/bundle.js",
```

```
"build:prod": "webpack --mode production --entry ./src/js/main.js --output  
./dist/bundle.min.js"
```

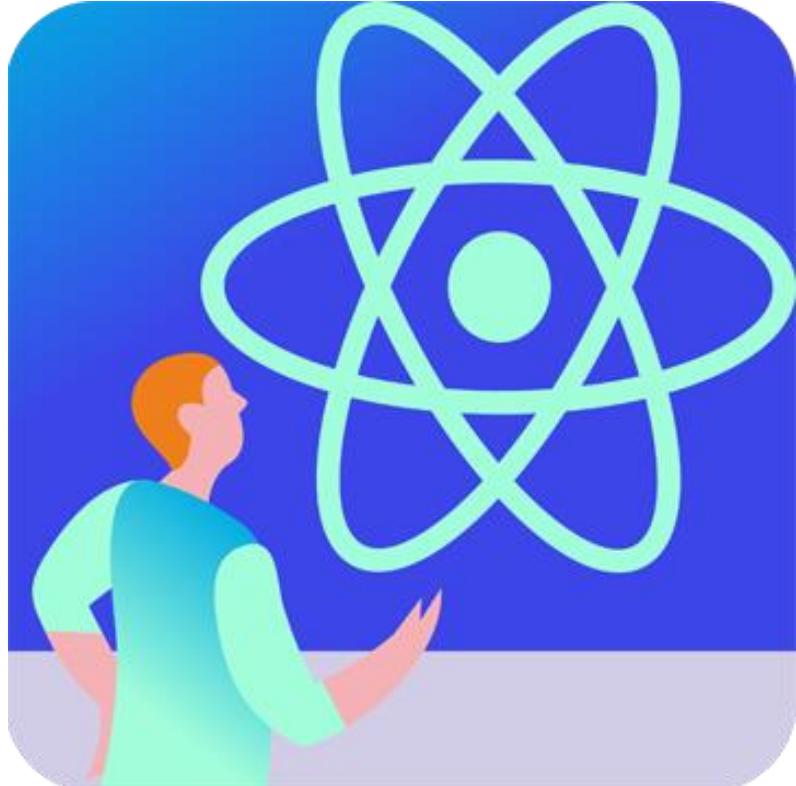
step 4: run เพื่อ build

npm run build

ถ้าต้องการ build เป็น production

npm run build:prod

step 5: แก้ลิงก์ js หน้า index เป็นไฟล์ dist/bundle.js



## Section 3

# บททวนพื้นฐาน React.JS

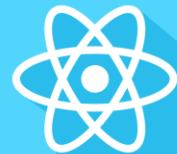
- React.js คืออะไร
- ติดตั้งเครื่องมือบน Google Chrome สำหรับใช้พัฒนา React
- สำรวจเว็บไซต์ที่พัฒนาด้วย React
- การ Render ด้วย React อย่างมีประสิทธิภาพ
- อัพเดต React เป็นรุ่นล่าสุด
- ติดตั้งเครื่องมือสำหรับการพัฒนา React.js

H E L L O !

รู้จัก React



# React คืออะไร



# React

- Javascript Library
- SPA
- Component

**React** เป็น Javascript Library หรือจะเรียกว่าเป็น Javascript Framework ก็ได้ ที่เราใช้สำหรับสร้างหน้าเว็บ พร้อมด้วย action ต่างๆ ที่ทำให้เว็บของเรารุดหน้าสบายนิ่ง

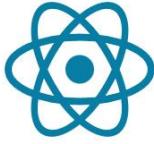
จุดเด่นของ React ที่ทำให้มันน่านำมาใช้งานนั้นก็คือ มันมีระบบแคลชในตัวทำให้หน้าเว็บของเราเมื่อการตอบสนองที่เร็ว เหราะแก่การนำไปทำ SPA เป็นอย่างยิ่ง

การเขียน React เราจึงสามารถแยกองค์ประกอบของหน้าเว็บเรา ออกเป็นส่วนๆ เรียกว่าเป็น component และนำมาประกอบกันเป็นหน้าเว็บได้ ซึ่งทำให้เราสามารถนำ component ของเราไปใช้ซ้ำที่อื่นได้ ไม่ต้องเสียเวลาเขียนใหม่

React react.dev

Support Ukraine UA Help Provide Humanitarian Aid to Ukraine.

Search Ctrl K Learn Reference Community Blog



# React

The library for web and native user interfaces

Learn React API Reference

## Create user interfaces

# FRONT-END FRAMEWORKS

ນີ້ມາ <https://2022.stateofjs.com/en-US/libraries/front-end-frameworks/>

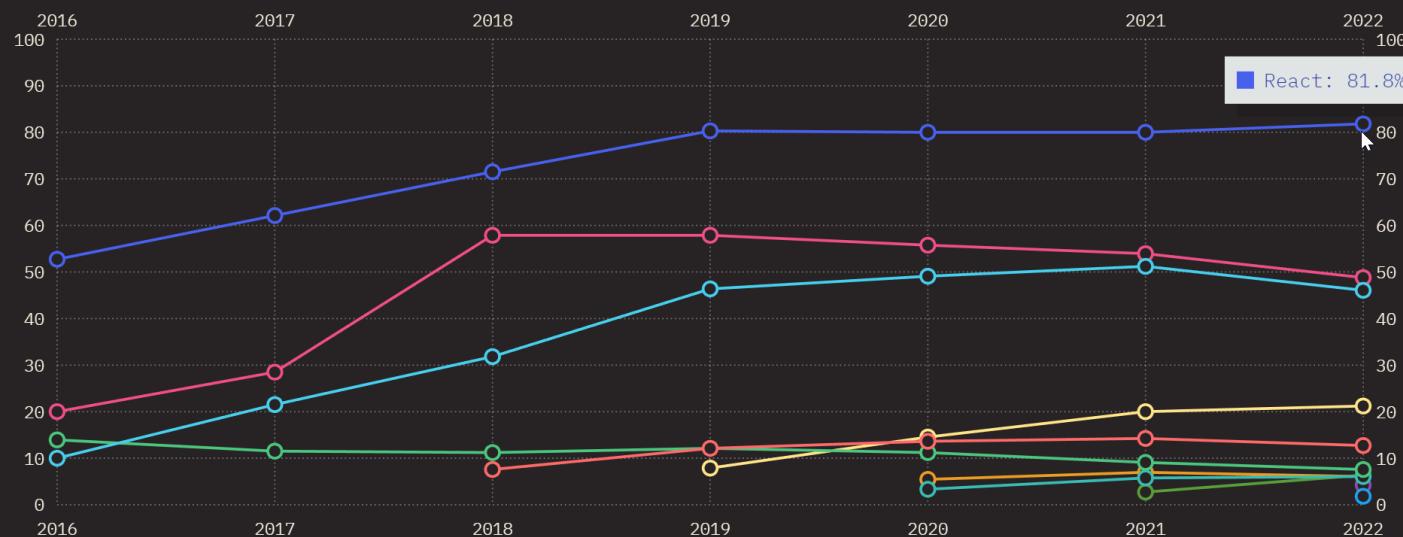
Front-end frameworks and libraries

## RATIOS OVER TIME

Percentages

Rankings

Retention, interest, usage, and awareness ratio over time.



Awareness   Usage   Interest   Retention

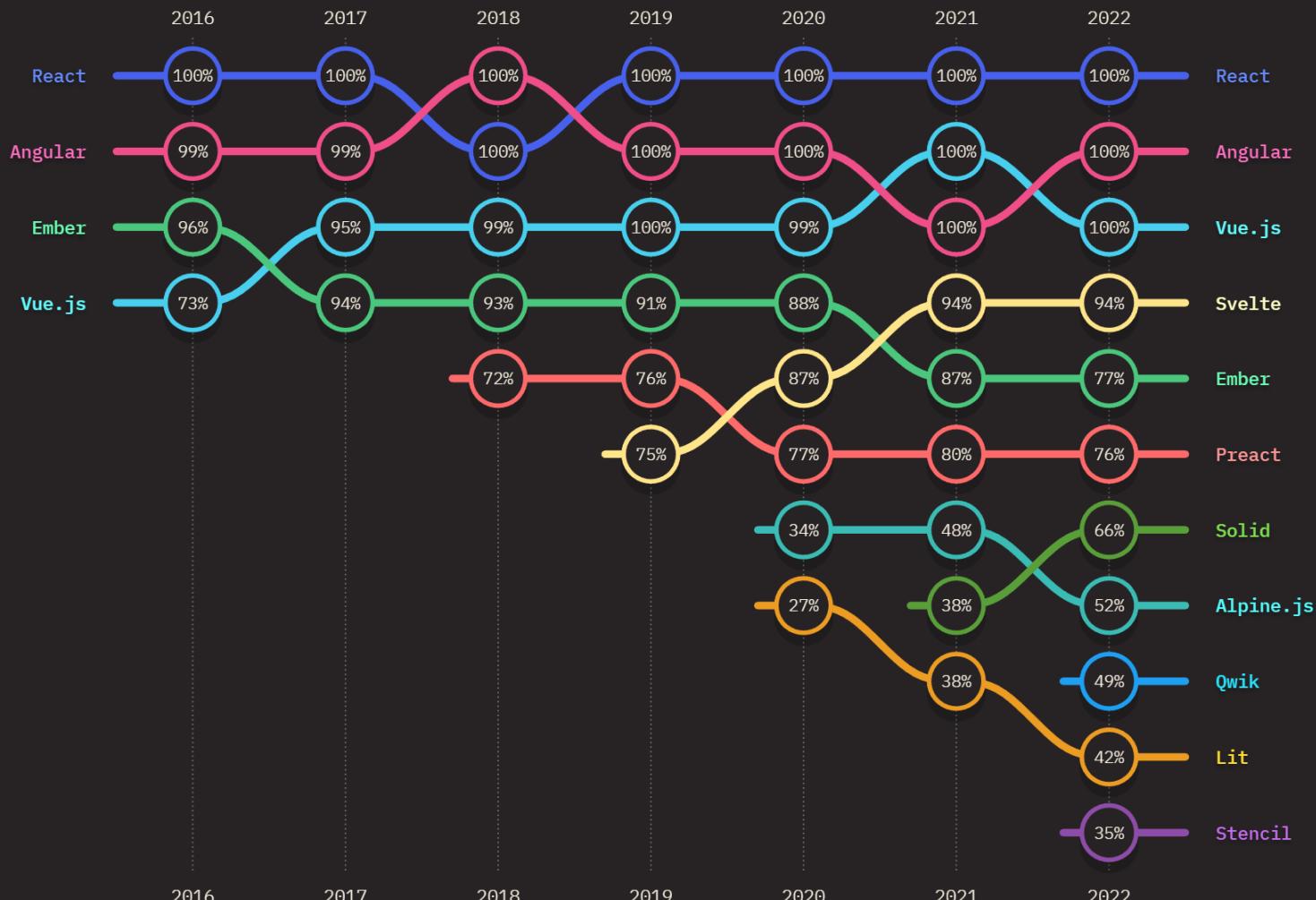
# RATIOS OVER TIME

\$

Percentages

Rankings

Retention, interest, usage, and awareness ratio rankings.  
కీమా <https://2022.stateofjs.com/en-US/libraries/front-end-frameworks/>



Awareness   Usage   Interest   Retention

# สามารถใช้ React สร้างสรรค์ผลงานแบบไหนได้บ้าง ?

- Single Page Application (SPA)
- Cross-platform mobile applications (with React Native)
- Dashboard or Data Visualization tools
- eCommerce or retail websites
- Enterprise Web Apps
- Messaging app
- Personal or professional blogs (with Gatsby)
- Social network

# สำรวจเว็บไซต์ที่พัฒนาด้วย React



# Companies Using React

**facebook**

**YAHOO!**

 **airbnb**



**The New York Times**

 **Dropbox**

The Dropbox logo features a blue diamond pattern consisting of six smaller diamonds forming a larger hexagon shape, followed by the word "Dropbox" in a bold, black, sans-serif font.

**NETFLIX**

 **Khan Academy**

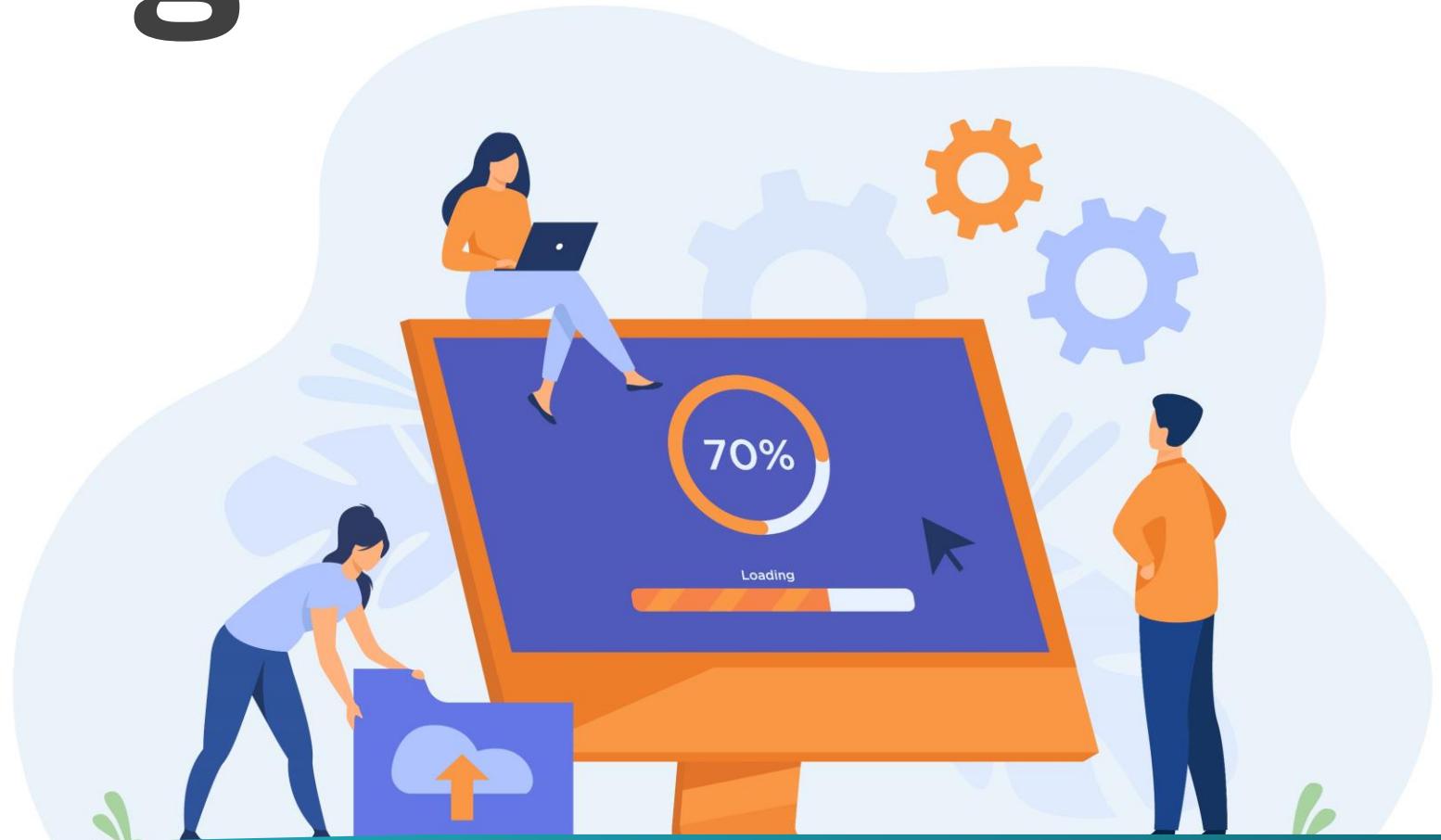
The Khan Academy logo features a green hexagon icon with a white stylized human figure inside, followed by the word "Khan Academy" in a dark blue, sans-serif font.

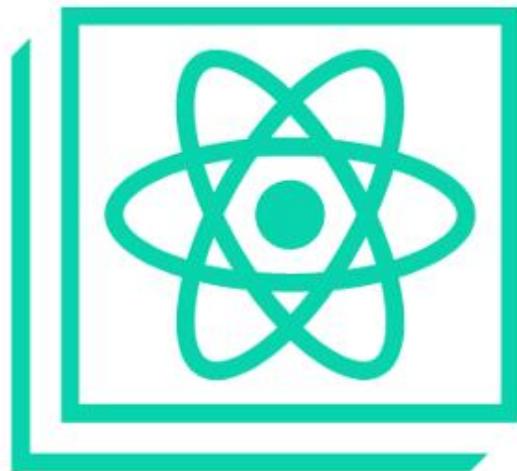
# Start

# React



# Installing React





## Create React App

Set up a modern web app by  
running one command.



## Vite

Next generation frontend tooling

## Create React App

Create React App is a comfortable environment for **learning React**, and is the best way to start building **a new single-page application** in React.

It sets up your development environment so that you can use the latest JavaScript features, provides a nice developer experience, and optimizes your app for production. You'll need to have `Node >= 14.0.0` and `npm >= 5.6` on your machine. To create a project, run:

```
npx create-react-app my-app  
cd my-app  
npm start
```

### Note

`npx` on the first line is not a typo — it's a package runner tool that comes with `npm 5.2+`.

Create React App doesn't handle backend logic or databases; it just creates a frontend build pipeline, so you can use it with any backend you want. Under the hood, it uses `Babel` and `webpack`, but you don't need to know anything about them.

When you're ready to deploy to production, running `npm run build` will create an optimized build of your app in the `build` folder. You can learn more about Create React App from its [README](#) and the [User Guide](#).

**Guide**

- Why Vite
- Getting Started
- Features
- Using Plugins
- Dependency Pre-Bundling
- Static Asset Handling
- Building for Production
- Deploying a Static Site
- Env Variables and Modes
- Server-Side Rendering (SSR)
- Backend Integration
- Comparisons
- Troubleshooting
- Migration from v2

**APIs**

- Plugin API
- HMR API
- JavaScript API
- Config Reference

 Search CtrlK[Guide](#) [Config](#) [Plugins](#) [Resources](#) [Version](#)

## Trying Vite Online

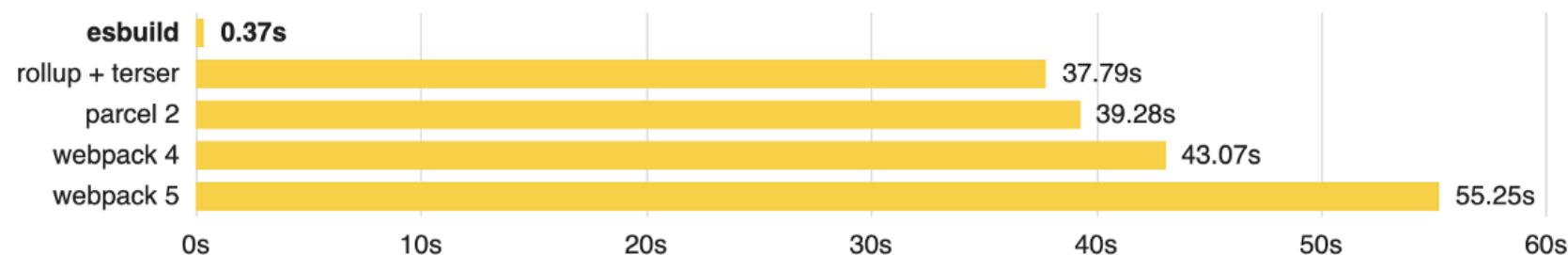
You can try Vite online on [StackBlitz](#). It runs the Vite-based build setup directly in the browser, so it is almost identical to the local setup but doesn't require installing anything on your machine. You can navigate to `vite.new/{template}` to select which framework to use.

The supported template presets are:

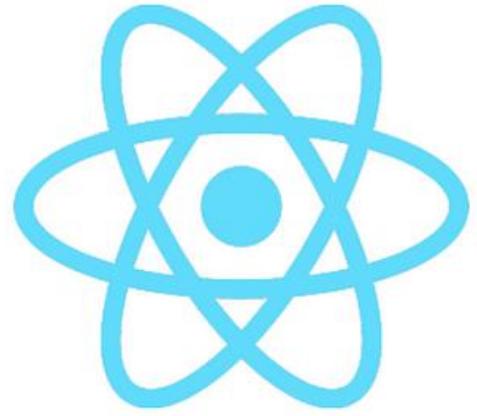
JavaScript	TypeScript
vanilla	vanilla-ts
vue	vue-ts
react	react-ts
preact	preact-ts
lit	lit-ts
svelte	svelte-ts

# esbuild

*An extremely fast JavaScript bundler*



*Above: the time to do a production bundle of 10 copies of the [three.js](#) library from scratch using default settings, including minification and source maps. More info [here](#).*



**React + Vite + TypeScript**



## New React plugin using SWC during development

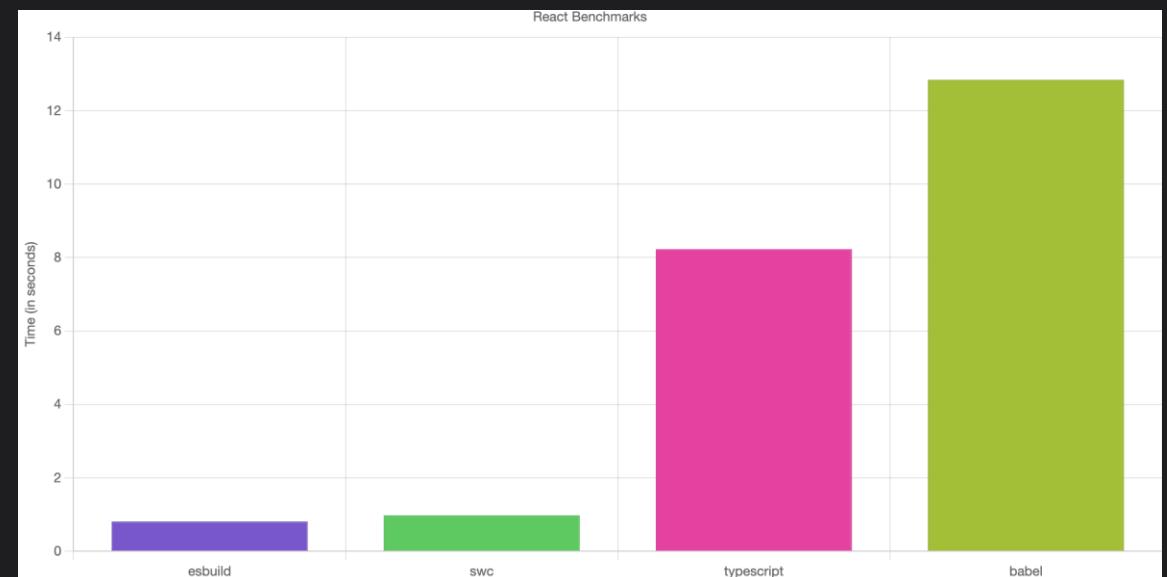
[SWC](#) is now a mature replacement for [Babel](#), especially in the context of React projects. SWC's React Fast Refresh implementation is a lot faster than Babel, and for some projects, it is now a better alternative. From Vite 4, two plugins are available for React projects with different tradeoffs. We believe that both approaches are worth supporting at this point, and we'll continue to explore improvements to both plugins in the future.

### @vitejs/plugin-react

[@vitejs/plugin-react](#) is a plugin that uses esbuild and Babel, achieving fast HMR with a small package footprint and the flexibility of being able to use the Babel transform pipeline.

### @vitejs/plugin-react-swc (new)

[@vitejs/plugin-react-swc](#) is a new plugin that uses esbuild during build, but replaces Babel with SWC during development. For big projects that don't require non-standard React extensions, cold start and Hot Module Replacement (HMR) can be significantly faster.



# Getting Started with Vite

## สร้างโปรเจกต์ด้วย npm

```
$ npm create vite@latest
```

```
• ReactNodeDockerClass ➔ npm create vite@latest
  ✓ Project name: ... sample-react-vite
  ✓ Select a framework: » React
  ✓ Select a variant: » TypeScript + SWC

Scaffolding project in G:\TrainingWorkshop\ReactNodeDockerClass\sample-react-vite...
Done. Now run:

  cd sample-react-vite
  npm install
  npm run dev
```

## ติดตั้ง Node dependencies ด้วย npm

```
$ npm install
```

## สร้างโปรเจกต์ด้วย yarn

```
$ yarn create vite
```



React + Vite + TypeScript

## ติดตั้ง Node dependencies ด้วย yarn

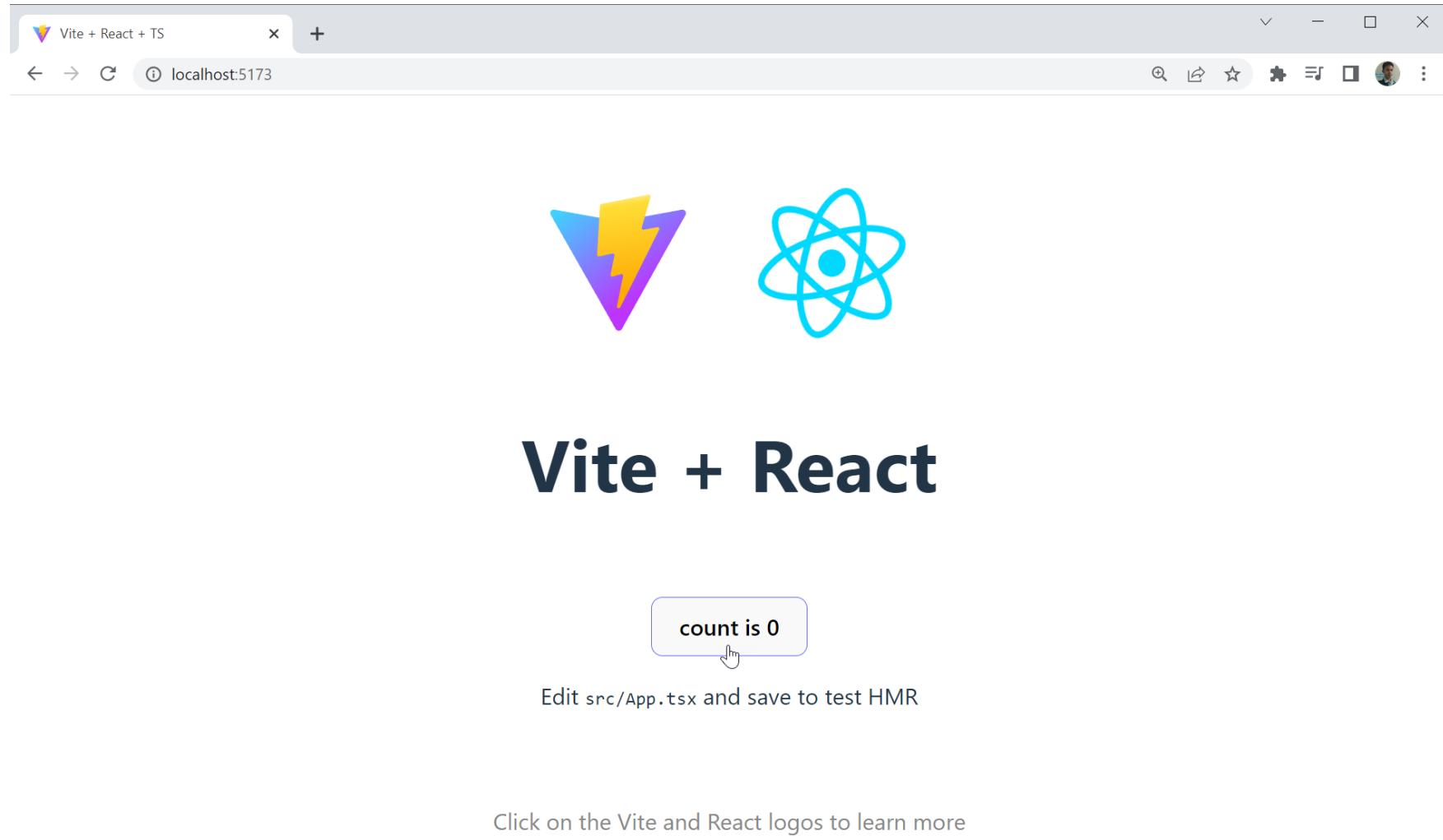
```
$ yarn install
```

# RUN REACT



- > **npm run dev**
- > **yarn dev**

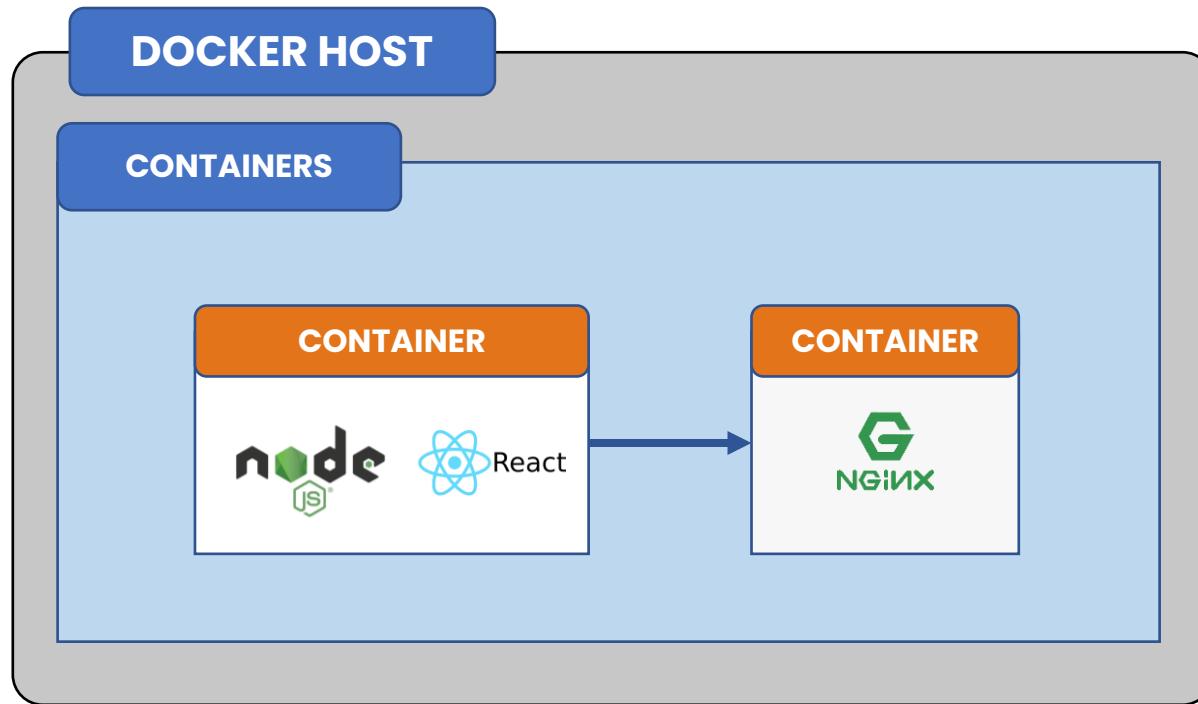
# Run React Project



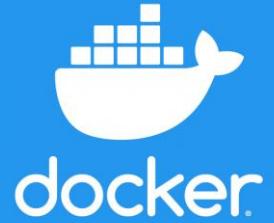


# React + Vite + Docker

# React Docker Stack



# Dockerfile



```
# Pull the base image
FROM node:18.16.0-alpine

# Set the working directory
WORKDIR /usr/app

# Copy app dependencies to container
COPY ./package*.json .

# Install dependencies
RUN npm install

# Copy code from host to container
COPY ..

# Expose Port
EXPOSE 5173

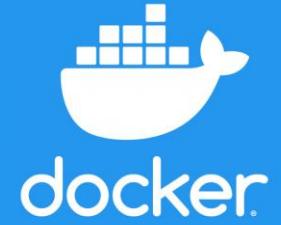
# Deploy app for local development
CMD [ "npm","run","dev" ]
```

# docker-compose.yml

```
version: '3.9'

# Network
networks:
  web_network:
    name: reactdockervite
    driver: bridge

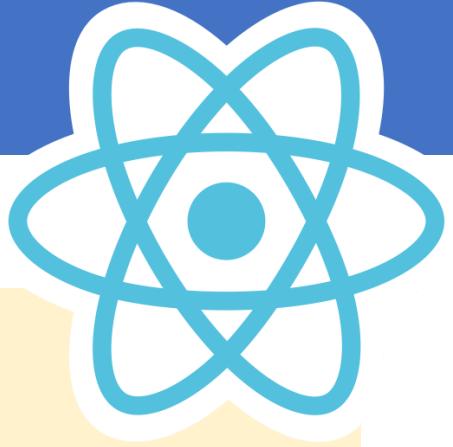
services:
  # React App Service
  reactapp:
    build:
      context: .
      dockerfile: Dockerfile
    container_name: reactapp_vite
    restart: always
    volumes:
      - .:/usr/app
      - /usr/app/node_modules
    ports:
      - 5173:5173
    environment:
      - CHOKIDAR_USEPOLLING=true
    networks:
      - web_network
```



# vite.config.js

```
import { defineConfig } from 'vite'
import react from '@vitejs/plugin-react'

// https://vitejs.dev/config/
export default defineConfig({
  plugins: [react()],
  server: {
    watch: {
      usePolling: true,
    },
    host: true,
    strictPort: true,
    port: 5173,
  }
})
```



# Command for Docker



## Check valid docker-compose file

```
$ docker-compose config -q
```

## Create Container

```
$ docker-compose up -d
```

## Remove Container

```
$ docker-compose down --rmi all
```

## Follow Logs in container

```
$ docker-compose logs -f [service_name]
```

## List Container Status

```
$ docker-compose ps
```

## Stop/Start Container

```
$ docker-compose stop/start
```

## Restart Container

```
$ docker-compose restart
```

## Exec to container

```
$ docker exec -it [container_name] sh
```

# โครงสร้าง โปรเจกต์ React



The screenshot shows the VS Code interface with the Explorer and Editor panes visible.

**Explorer:** Shows the project structure:

- node\_modules
- public
  - vite.svg
- src
  - assets
    - react.svg
  - App.css
  - App.tsx
  - index.css
  - main.tsx
  - vite-env.d.ts
- .eslintrc.cjs
- .gitignore
- index.html
- package-lock.json
- package.json
- tsconfig.json
- tsconfig.node.json
- vite.config.ts

**Editor:** Displays the contents of the package.json file:

```
1  {
2    "name": "sample-react-vite",
3    "private": true,
4    "version": "0.0.0",
5    "type": "module",
6    "scripts": {
7      "dev": "vite",
8      "build": "tsc && vite build",
9      "lint": "eslint src --ext ts,tsx
--report-unused-disable-directives --max-warnings 0",
10     "preview": "vite preview"
11   },
12   "dependencies": {
13     "react": "^18.2.0",
14     "react-dom": "^18.2.0"
15   },
16   "devDependencies": {
17     "@types/react": "^18.0.28",
18     "@types/react-dom": "^18.0.11",
19     "@typescript-eslint/eslint-plugin": "^5.57.1",
20     "@typescript-eslint/parser": "^5.57.1",
21     "@vitejs/plugin-react-swc": "^3.0.0", I
22     "eslint": "^8.38.0",
23     "eslint-plugin-react-hooks": "^4.6.0",
24     "eslint-plugin-react-refresh": "^0.3.4",
25     "typescript": "^5.0.2",
26     "vite": "^4.3.2"
27   }
28 }
29 }
```

The screenshot shows a code editor interface with a dark theme. On the left is a sidebar displaying the project structure:

- > node\_modules
- public
  - vite.svg
- src
  - assets
    - react.svg
  - App.css
  - App.tsx
  - index.css
  - main.tsx
  - vite-env.d.ts
- .eslintrc.cjs
- .gitignore
- index.html
- package-lock.json
- package.json
- tsconfig.json
- tsconfig.node.json
- vite.config.ts

The main editor area shows the content of the App.tsx file:

```
1 import { useState } from 'react'
2 import reactLogo from './assets/react.svg'
3 import viteLogo from '/vite.svg'
4 import './App.css'
5
6 function App() {
7   const [count, setCount] = useState(0)
8
9   return (
10    <>
11      <div>
12        <a href="https://vitejs.dev" target="_blank">
13          <img src={viteLogo} className="logo" alt="Vite logo" />
14        </a>
15        <a href="https://react.dev" target="_blank">
16          <img src={reactLogo} className="logo react" alt="React logo" />
17        </a>
18      </div>
19      <h1>Vite + React + TypeScript + SWC</h1>
20      <div className="card">
21        <button onClick={() => setCount((count) => count + 1)}>
22          count is {count}
23        </button>
24        <p>
25          Edit <code>src/App.tsx</code> and save to test HMR
26        </p>
27      </div>
28      <p className="read-the-docs">
29        Click on the Vite and React logos to learn more
30      </p>
31    </>
32  }
33 }
```

EXP... ⌂ ⌂ ⌂ ⌂ ...

main.tsx X

> node\_modules

✓ public

\* vite.svg

✓ src

✓ assets

\* react.svg

App.css

App.tsx

index.css

main.tsx

vite-env.d.ts

.eslintrc.cjs

.gitignore

index.html

package-lock.json

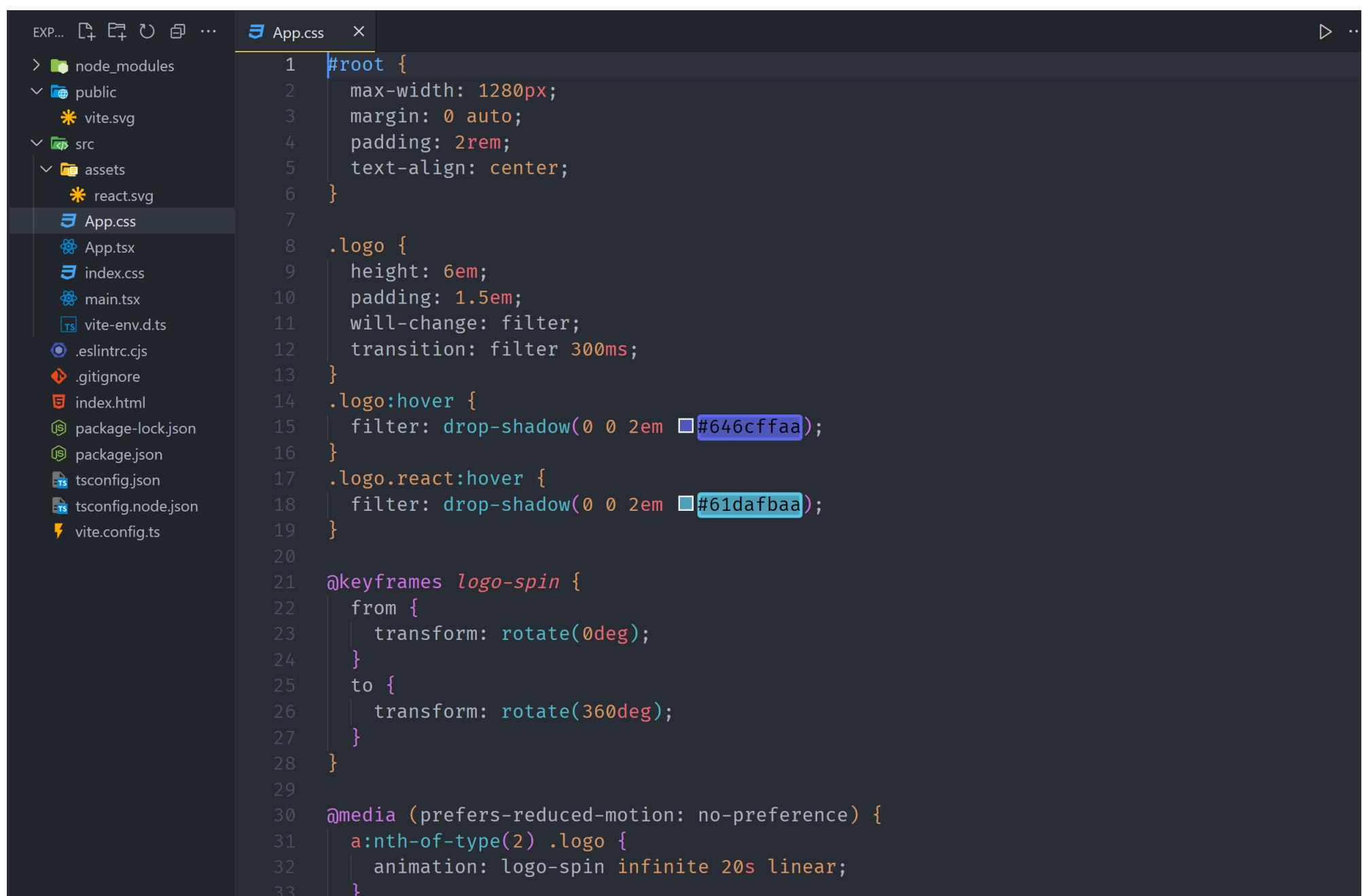
package.json

tsconfig.json

tsconfig.node.json

vite.config.ts

```
1 import React from 'react'
2 import ReactDOM from 'react-dom/client'
3 import App from './App.tsx'
4 import './index.css'
5
6 ReactDOM.createRoot(document.getElementById('root') as HTMLElement).render(
7   <React.StrictMode>
8     <App />
9   </React.StrictMode>,
10 )
11 
```



The screenshot shows a code editor interface with a dark theme. The left sidebar displays a file tree for a project structure:

- > node\_modules
- ✓ public
  - \* vite.svg
- ✓ src
  - assets
    - \* react.svg
  - App.css
  - App.tsx
  - index.css
  - main.tsx
  - vite-env.d.ts
  - .eslintrc.cjs
  - .gitignore
  - index.html
  - package-lock.json
  - package.json
  - tsconfig.json
  - tsconfig.node.json
  - vite.config.ts

The screenshot shows a code editor interface with a dark theme. The left sidebar displays a file tree for a project structure:

- > node\_modules
- public
  - vite.svg
- src
  - assets
    - react.svg
  - App.css
  - App.tsx
  - index.css
- main.tsx
- vite-env.d.ts
- .eslintrc.cjs
- .gitignore
- index.html
- package-lock.json
- package.json
- tsconfig.json
- tsconfig.node.json
- vite.config.ts

```
1  :root {  
2    font-family: Inter, system-ui, Avenir, Helvetica, Arial, sans-serif;  
3    line-height: 1.5;  
4    font-weight: 400;  
5  
6    color-scheme: light dark;  
7    color: #rgba(255, 255, 255, 0.87);  
8    background-color: #242424;  
9  
10   font-synthesis: none;  
11   text-rendering: optimizeLegibility;  
12   -webkit-font-smoothing: antialiased;  
13   -moz-osx-font-smoothing: grayscale;  
14   -webkit-text-size-adjust: 100%;  
15 }  
16  
17 a {  
18   font-weight: 500;  
19   color: #646cff;  
20   text-decoration: inherit;  
21 }  
22 a:hover {  
23   color: #535bf2;  
24 }  
25  
26 body {  
27   margin: 0;  
28   display: flex;  
29   place-items: center;  
30   min-width: 320px;  
31   min-height: 100vh;  
32 }
```

The screenshot shows a code editor interface with a dark theme. On the left is a sidebar displaying the project file structure:

- > node\_modules
- ✓ public
  - \* vite.svg
- ✓ src
  - assets
    - \* react.svg
  - App.css
  - App.tsx
  - index.css
  - main.tsx
  - vite-env.d.ts
- .eslintrc.cjs
- .gitignore
- ✓ index.html
- package-lock.json
- package.json
- tsconfig.json
- tsconfig.node.json
- vite.config.ts

The main editor area is titled "index.html" and contains the following code:

```
1  <!DOCTYPE html>
2  <html lang="en">
3      <head>
4          <meta charset="UTF-8" />
5          <link rel="icon" type="image/svg+xml" href="/vite.svg" />
6          <meta name="viewport" content="width=device-width, initial-scale=1.0" />
7          <title>Vite + React + TS</title>
8      </head>
9      <body>
10         <div id="root"></div>
11         <script type="module" src="/src/main.tsx"></script>
12     </body>
13 </html>
```

EXP... ⌂ ⌂ ⌂ ⌂ ...

tsconfig.json X

> node\_modules  
✓ public vite.svg  
✓ src assets react.svg  
App.css  
App.tsx  
index.css  
main.tsx  
vite-env.d.ts  
.eslintrc.cjs  
.gitignore  
index.html  
package-lock.json  
package.json  
tsconfig.json  
tsconfig.node.json  
vite.config.ts

```
1  {  
2      "compilerOptions": {  
3          "target": "ESNext",  
4          "lib": ["DOM", "DOM.Iterable", "ESNext"],  
5          "module": "ESNext",  
6          "skipLibCheck": true,  
7  
8          /* Bundler mode */  
9          "moduleResolution": "bundler",  
10         "allowImportingTsExtensions": true,  
11         "resolveJsonModule": true,  
12         "isolatedModules": true,  
13         "noEmit": true,  
14         "jsx": "react-jsx",  
15  
16         /* Linting */  
17         "strict": true,  
18         "noUnusedLocals": true,  
19         "noUnusedParameters": true,  
20         "noFallthroughCasesInSwitch": true  
21     },  
22     "include": ["src"],  
23     "references": [{ "path": "./tsconfig.node.json" }]  
24 }
```

```
1 import { defineConfig } from 'vite'
2 import react from '@vitejs/plugin-react-swc'
3
4 // https://vitejs.dev/config/
5 export default defineConfig({
6   plugins: [react()],
7 })
8
```

The screenshot shows a Visual Studio Code (VS Code) interface with the following details:

- EXPLORER** sidebar:
  - SAMPLE-REACT-VITE folder structure:
    - node\_modules
    - public
      - vite.svg
    - src
      - assets
        - react.svg
      - App.css
      - App.tsx
      - index.css
    - main.tsx
    - vite-env.d.ts
  - .eslintrc.cjs
  - .gitignore
  - index.html
  - package-lock.json
  - package.json
  - tsconfig.json
  - tsconfig.node.json
  - vite.config.ts
- Code Editor**:
  - main.tsx (top pane):

```
1 import React from 'react'
2 import ReactDOM from 'react-dom/client'
3 import App from './App.tsx'
4 import './index.css'

6 ReactDOM.createRoot(document.getElementById('root') as HTMLElement).render(
7   <React.StrictMode>
8     | <App />
9   </React.StrictMode>,
10 )
```
  - App.tsx (second pane):

```
1 import { useState } from 'react'
2 import reactLogo from './assets/react.svg'
3 import viteLogo from '/vite.svg'
4 import './App.css'

6 function App() {
7   const [count, setCount] = useState(0)

9   return (
10     <>
11       <div>
12         <a href="https://vitejs.dev"
13           target="_blank">
14           <img src={viteLogo}
```
  - App.css (third pane):

```
1 #root {
2   max-width: 1280px;
3   margin: 0 auto;
4   padding: 2rem;
5   text-align: center;
6 }

8 .logo {
9   height: 6em;
10  padding: 1.5em;
11  will-change: filter;
12  transition: filter 300ms;
13 }
14 .logo:hover {
```
  - index.css (fourth pane):

```
1 :root {
2   font-family: Inter,
3   system-ui, Avenir, Helvetica,
4   Arial, sans-serif;
5   line-height: 1.5;
6   font-weight: 400;
7   color-scheme: light dark;
8   color: □rgba(255, 255, 255, 0.
87);
9   background-color: ■#242424;
10  font-synthesis: none;
11  text-rendering:
12  optimizeLegibility;
13  -webkit-font-smoothing:
14    antialiased;
15  -moz-osx-font-smoothing:
16    grayscale;
17  -webkit-text-size-adjust:
18    100%;
```
  - index.html (fifth pane):

```
1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="UTF-8" />
5     <link rel="icon" type="image/svg+xml" href="/vite.svg" />
6     <meta name="viewport" content="width=device-width,
7     initial-scale=1.0" />
8     <title>Vite + React + TS</title>
9   </head>
10  <body>
11    <div id="root"></div>
12    <script type="module" src="/src/main.tsx"></script>
13  </body>
14 </html>
```
- Bottom Status Bar**: Go 1.17.3, X 0 △ 0, (.) 6, β, Ln 1, Col 26, Spaces: 2, UTF-8, LF, {}, TypeScript JSX, ▲ Go Update Available, ▵ Prettier, ▷, ▷



## แนะนำ JSX

- ทำไมต้องใช้ JSX
- ข้อกำหนดพื้นฐานของ JSX
- การแสดงค่าของตัวแปรใน JSX
- การคำนวณค่าในและแสดงผล (Expression) ใน JSX
- การแสดงหรือซ่อนเนื้อหาใน JSX

# ทำไมต้อง ใช้ JSX



# What Is JSX?

JSX คือ ส่วนขยายของ JavaScript (JavaScript Extension) ที่ใช้สำหรับสร้างส่วนติดต่อกับผู้ใช้ (UI) โดย JSX มีลักษณะคล้ายกับโค้ด HTML แต่มีกฎเกณฑ์บางอย่างที่แตกต่างจาก HTML

HTML

```
<div class="myElement">This is HTML Element</div>
```

JSX

```
<App className="App">This is JSX Element</App>
```

# ข้อกำหนดของ JSX

## JSX ทั้งหมดต้องอยู่ภายใต้ Element เดียว (Single Parent)

✓ Right

```
function App(){
  return (
    <div className="App">
      <div>เนื้อหาส่วนที่ 1</div>
      <div>เนื้อหาส่วนที่ 2</div>
    </div>
  )
}
```

✗ Wrong

```
function App(){
  return (
    <div className="App">
      <div>เนื้อหาส่วนที่ 1</div>
    </div>
    <div className="App">
      <div>เนื้อหาส่วนที่ 2</div>
    </div>
  )
}
```

# การแสดงค่าของตัวแปรใน JSX

```
function App(){
  const myName = 'Samit Koyom'
  return (
    <div className="App">
      <h1>ชื่อของพมคือ { myName }</h1>
    </div>
  )
}
```

# การแสดงค่าจากการคำนวณ (Expression) ใน JSX

```
function App(){
  const mySalary = 24000
  const myWifeSalary = 16000
  return (
    <div className="App">
      <h1>รายได้รวมของครอบครัว { mySalary + myWifeSalary }</h1>
    </div>
  )
}
```

# การแสดงหรือซ่อนเนื้อหาในคอมโพบเนต

```
function App(){
  const randomNumber = Math.random()
  return (
    <div className="App">
      <p>Random value: { randomNumber }</p>
      {
        randomNumber < 0.5 ? <div>คุณแพ้</div> : <div>คุณชนะ</div>
      }
    </div>
  )
}
```

# Return โค้ดหลายบรรทัดต้องใส่วงเล็บ

```
function App(){  
  return <div className="App">Live สดถือทะเบียนสมรสไปงานแต่ง</div>  
}
```

```
function App(){  
  return (  
    <div className="App">  
      <h1>Live สดถือทะเบียนสมรสไปงานแต่ง</h1>  
      <h2>ล่าสุดพ้องศาลแล้ว เรียกเงินจากเจ้าสาว</h2>  
    </div>  
  )  
}
```

# Styles JSX



# Styled JSX

สามารถใส่ CSS ลงไปใน JSX ได้ 2 แบบด้วยกันคือ

1. กำหนดแบบ Inline CSS
2. กำหนดแบบ Link Style

# Inline CSS

```
style={{color:'green', fontSize:'24'}}
```

```
<div style={{color:'green', fontSize:'24'}}>เวลาเป็นเครื่องพิสูจน์ ให้มากรูดเป็นเครื่องตั้มยำ</div>
```

# Inline CSS

```
<p style={  
  color:'blue',  
  fontSize: 18,  
  border:'1px solid red',  
  textAlign:'left',  
  padding: 20  
}>  
  "มีตրภาพ" ไม่ได้เริ่มจากวงเหล้า "มีตրภาพ" ไม่ได้เริ่มจากวงไฟ <br />  
  "มีตրภาพ" ไม่ได้เริ่มจากวงหมอลำ <br />  
  ||ต่จากการศึกษาค้นคว้า ของหลาย ๆ สถาบัน สรุปตรงกันว่า <br />  
  "มีตրภาพ" เริ่มจากสระบูรี และ ไปสืบสุดที่หนองคาย <br />  
  จะนั้น...อย่าไปหนองคาย <br />  
  มีตรภาพ...จะสืบสุดกันที่ <br />  
</p>
```

# การกำหนด CSS แบบ Link Style

## src/App.css

```
.myStyle {  
    color: teal;  
    font-size: 30px;  
    border: 2px solid #61dafb;  
    border-radius: 10px;  
    padding: 10px;  
    text-align: left;  
}
```

# การกำหนด CSS แบบ Link Style

```
import React from 'react'
import './App.css'

function App(){
  return (
    <div className="App">
      <p className="myStyle">
        เรอบัญชีอะไรจันจะโวน<br />
        กິ່ງหน้าຕາຂອງຈັນເມື່ອນກັນໂຈຣ<br />
        ເບວົດເຮົວເບວົດວະໄຮຈັນຈະໂກຣ<br />
        ອຍ່າກຳໃຫ້ຈັນເໜີງ ເໜີງ Alone
      </p>
    </div>
  )
}
```

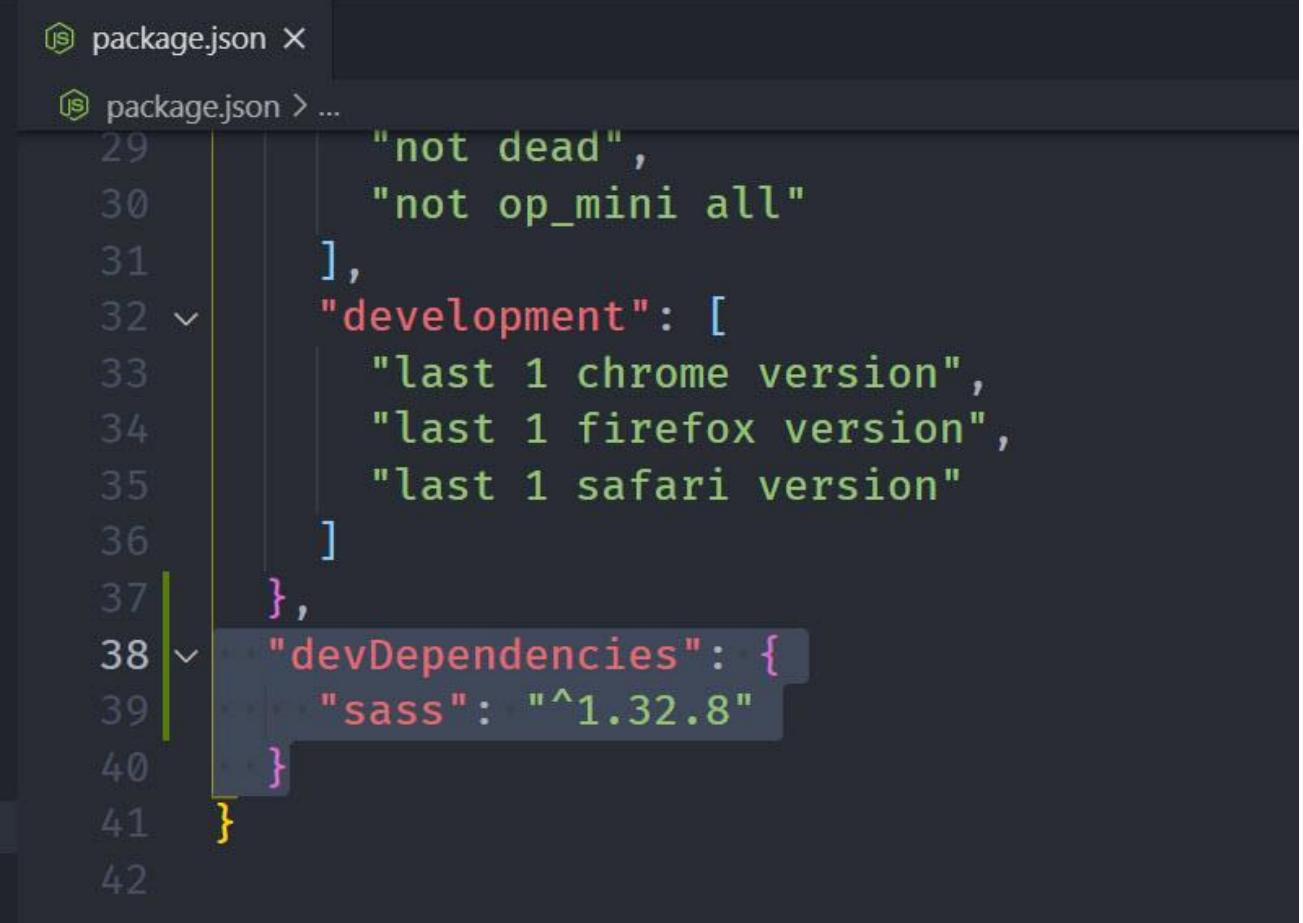


# Sass Support

# ติดตั้ง Sass

npm install sass --save-dev

npm install sass -D



The screenshot shows a code editor with a dark theme displaying a `package.json` file. The file contains the following JSON code:

```
 29     "not dead",
 30     "not op_mini all"
 31   ],
 32   "development": [
 33     "last 1 chrome version",
 34     "last 1 firefox version",
 35     "last 1 safari version"
 36   ],
 37 },
 38 "devDependencies": {
 39   "sass": "^1.32.8"
 40 }
 41 }
 42 }
```

The line `"sass": "^1.32.8"` is highlighted with a blue selection rectangle, indicating it is the current focus or selected for modification.

# สร้างไฟล์ Style.scss

```
.myStyle {  
color: teal;  
font-size: 30px;  
border: 2px solid #61dafb;  
border-radius: 10px;  
padding: 10px;  
text-align: left;  
}
```

# เรียนใช้งาน Style.scss

```
import React from 'react'  
import './Style.scss'  
  
function App(){  
  return (  
    <div>  
      <p>  
        คนจะเจ้าชู้ต้องดูที่อะไร<br />  
        คนจะหลายใจต้องดูอะไรบ้างหนา<br />  
        ดูที่รอยยิ้มหรือดูที่ใบหน้า<br />  
        ดูที่ดวงตาหรือดูที่อะไร  
      </p>  
    </div>  
  )  
}
```





## แนะนำ React Component

- คุณสมบัติพื้นฐานของคอมโพเนนต์
- วิธีสร้างคอมโพเนนต์ใช้เอง
- ยกตัวอย่างคลาสคอมโพเนนต์และฟังก์ชันคอมโพเนนต์และความแตกต่าง
- วิธีการสร้างของคอมโพเนนต์ใน React อย่างละเอียด
- การนำคอมโพเนนต์ไปใช้งาน
- เทคนิคการ reuse (ใช้ซ้ำ) ที่ดีในคอมโพเนนต์

# Creating components

NavBar

Profile  
Dashboard

Feed

Who to  
Follow

Trends

# React สามารถสร้าง Component ได้ 2 รูปแบบ

1. Class Component
2. Function Component

# React สามารถสร้าง Component ได้ 2 รูปแบบ

## 1. คลาสคอมโพเนนต์ (Class Component)

```
import React, { Component } from 'react';

class Post extends Component {
  render() {
    return (
      <div>
        <p>Hello Class Component!!!</p>
      </div>
    );
  }
}

export default Post;
```

# React สามารถสร้าง Component ได้ 2 รูปแบบ

## 2. พังก์ชันคอมโพเนนต์ (Function Component)

เขียนพังก์ชันแบบปกติ

```
function Post() {  
  return (  
    <div>  
      <p>Hello Function Component!!!</p>  
    </div>  
  )  
}  
  
export default Post
```

เขียนพังก์ชันแบบ Arrow (=>)

```
const Post = () => {  
  return (  
    <div>  
      <p>Hello Function Component!!!</p>  
    </div>  
  )  
}  
  
export default Post
```

# เปรียบเทียบ Class vs Function Component

## Class Component

```
import React, { Component } from 'react'

class Post extends Component {
  render() {
    return (
      <div>
        <p>Hello Class Component!!!</p>
      </div>
    );
  }
}

export default Post
```

## Functional Component

```
const Post = () => {
  return (
    <div>
      <p>Hello Function Component!!!</p>
    </div>
  )
}

export default Post
```



## การใช้งาน css และ Bootstrap 5 framework ใน React

- แนวทางการใช้ css ปรับแต่งหน้าตาคอมโพเนนต์
- วิธีกำหนด css แบบ inline
- การกำหนด css แบบ Link Style
- การใช้ Bootstrap 5 ปรับแต่งหน้าตาของ คอมโพเนนต์

# ติดตั้ง Bootstrap 5.x

```
npm install bootstrap@5.x
```

# React and NodeJS with Docker Workshop Progress



Day 1



Day 2



Day 3



Day 4



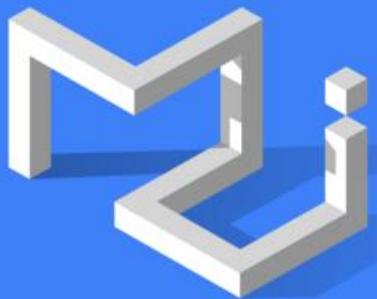
Day 5

# DAY 2

**Section 4:** สร้าง Workshop โปรเจกต์ React 19 ด้วย Vite

**Section 5:** ใช้ React ออกแบบ UI ด้วย Material UI (MUI)  
และ MUI X ล่าสุด





Material-UI

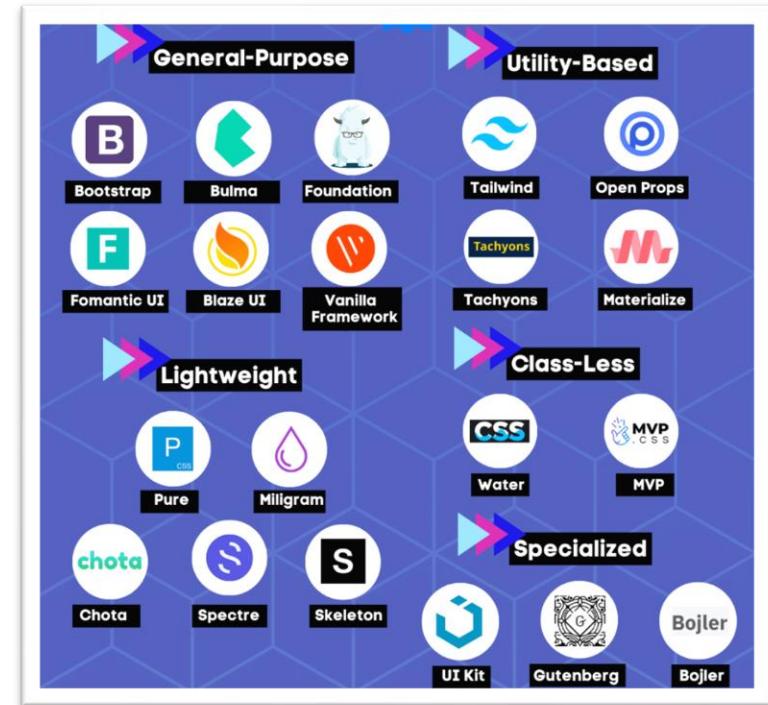
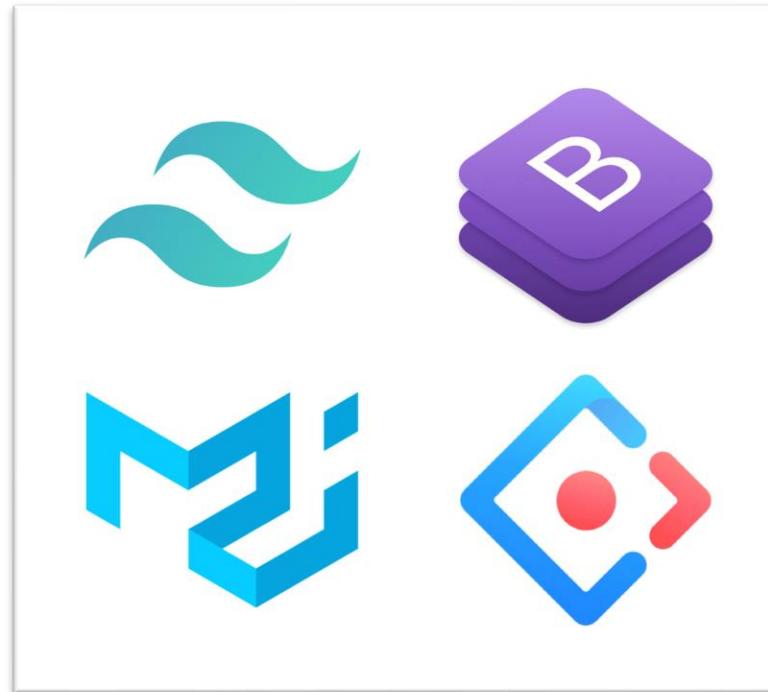
## การใช้งาน Material UI (MUI)

- MUI (Opensource CSS Theme) Integration
- Flexbox - Modern CSS for Web Responsive
- Add Components and Assets to MaterialUI
- Data Exchange between Components both (Child and Parent)
- Callback with props between Components
- Routing Solution (Route, Link, Redirect, Match Page not Found) for (SPA)

# What are CSS Frameworks ?

A CSS framework refers to a prepped-up library that contains ready-to-use CSS stylesheet collections, making the job of UI developers easier. It contains basic site structures such as **grids** and **flex boxes**, **text typography**, **buttons**, **icons**, and **interactive user interfaces**. It could also contain pre-built site components such as **modals**, **navbars**, and **cards**. Examples of popular CSS frameworks include

- [Bootstrap](#)
- [Tailwind CSS](#)
- [Material UI](#)
- [Foundation](#)
- [Pure](#)
- [Bulma](#)
- [Semantic UI](#)
- [UI kit](#)
- [Materialize CSS](#)
- [Spectre](#)
- [Base](#)
- [Picnic CSS](#)



# What is Material UI ?

Material UI is an open-source, front-end user interface library that features components for speeding up the creation of web applications, allowing the developer to save time on design and development. It can be used to add site essentials, such as **buttons**, **sliders**, **navigation**, **cards**, **grids**, etc.

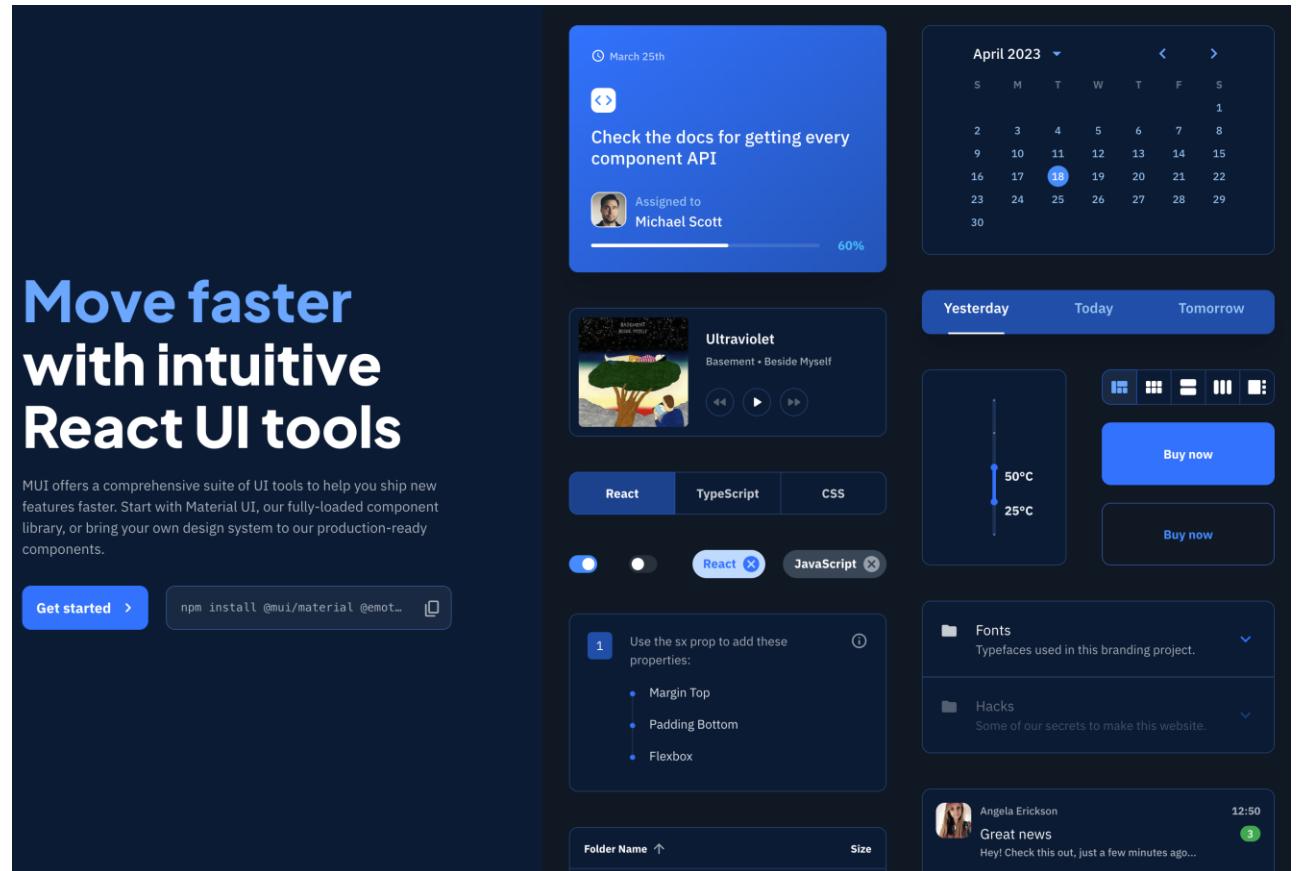


The screenshot displays the Material UI pricing page. At the top, there's a navigation bar with tabs for "Plans", "Community", "Pro", and "Premium". The "Community" tab is selected, showing the "Recommended" plan. Below this, the "Community" plan is listed with a price of \$0 and a note that it's free forever. The "Pro" plan is listed with a price of \$15 per developer per month, noting it's billed annually at \$180 per developer. The "Premium" plan is listed with a price of \$37 per developer per month, also noting it's billed annually at \$444 per developer. Each plan has a "Get started" button and a "Buy now" button. Below the plans, there are two sections: "MUI Core (open-source)" and "MUI X (open-core)". The "MUI Core" section lists components like Material UI, Joy UI, Base UI, and MUI System, each with a checkmark indicating they are included in all three plans. The "MUI X" section lists components like Data Grid, Date Picker, Date Range Picker, Perpetual use in production, Development license, and Access to new releases, with checkmarks indicating their availability across the different plan levels.

Component	Community (\$0)	Pro (\$15)	Premium (\$37)
Material UI	✓	✓	✓
Joy UI	✓	✓	✓
Base UI	✓	✓	✓
MUI System	✓	✓	✓
MUI X (open-core)			
Data Grid	↔	↔	↔
Date Picker	✓	✓	✓
Date Range Picker	-	✓	✓
Perpetual use in production	✓	✓	✓
Development license	✓	1 year	1 year
Access to new releases	✓	1 year	1 year

# Benefits of using Material-UI

- Well-designed responsive components, helping you to focus more on business logic.
- A large community and properly written document.
- Design consistency
- Faster development with ready-made components.



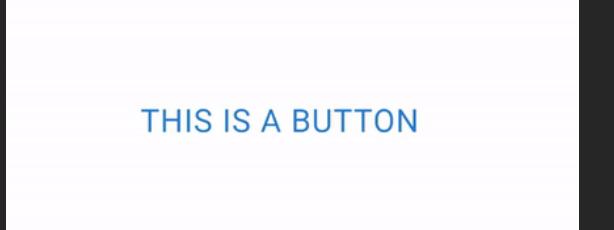
# Getting Started

```
npm install @mui/material @emotion/react @emotion/styled
```

## Using in App.js

```
import {Button} from '@mui/material'
```

```
function App() {
  return (
    <div className="App">
      <Button>This is a button</Button>
    </div>
  )
}
```



THIS IS A BUTTON

React Stock

Dashboard

OVERVIEW CONTENT AUDIENCE REVENUE

Samit Koyom  
Administrator

Dashboard Products Analytics Customization

Views 21.4K 700 less than usual

Watch Time 400 27 more than usual

Subscribers +140 about the same as usual

Revenue \$240.02 700 more than usual

Realtime  
Updating Live

4,144 Subscribers

1,786 Views Last 48 hours

See more

Jan 19, 2023 Jan 29, 2023 Feb 8, 2023

React Stock

Samit Koyom  
Administrator

Dashboard

Products

Analytics

Customization

## Products

ID	Category	Name	Price	Image	Qty	Create	Action
1	Electronic	iPhone 14 Pro Max	40000		2	2023-04-27T04:22:01	<button>EDIT</button> <button>DELETE</button>
2	Electronic	iPad Pro 2023	35000		5	2023-04-27T04:22:01	<button>EDIT</button> <button>DELETE</button>
3	Electronic	ลำโพงลู๊ฟพกพา	2999		10	2023-04-27T04:22:01	<button>EDIT</button> <button>DELETE</button>



# What are React.Js Props

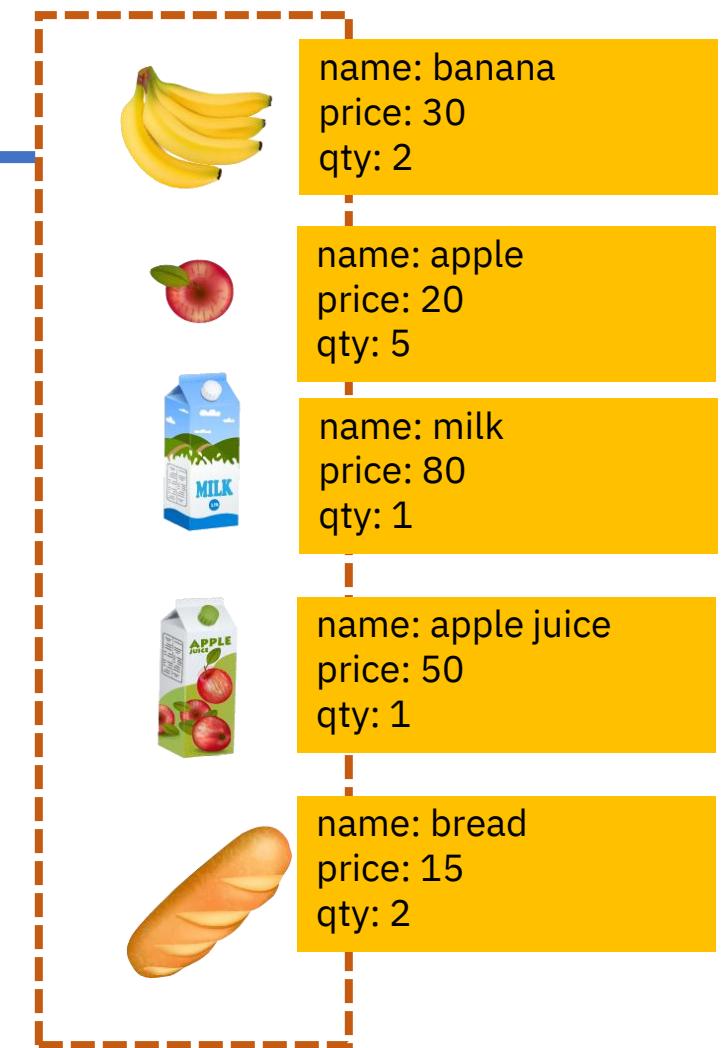
## การทำงานกับ Props

- รู้จักกับ props ใน React
- วิธีส่งค่าผ่าน props ไปยังคอมโพเนนต์ลูก (child)
- การส่งค่าหลายๆ ค่าไปยังคอมโพเนนต์ลูก (child)
- การนำเอาค่า props มาใช้ในคอมโพเนนต์ทั้งในแบบคลาสและฟังก์ชันคอมโพเนนต์
- การรับค่า children ที่มาระบุจาก props

# Cart

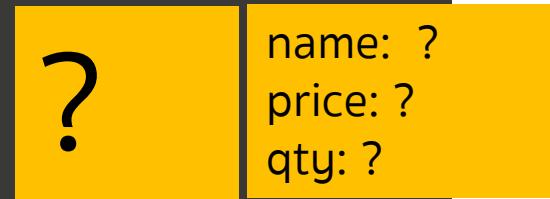


# Cart Items



## CartItems.jsx

```
const CartItems = (props) => {
  return (
    <div className="card">
      <p>Name: {props.name}</p>
      <p>Price: {props.price}</p>
      <p>Qty: {props.qty}</p>
    </div>
  )
}
```



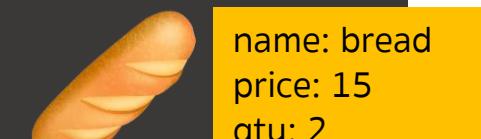
## Cart.jsx

```
import CartItems from "./CartItems"
```

```
const Cart = () => {
  return (
    <div className="card">
      <CartItems name="banana" price="30" qty="2" />
      <CartItems name="bread" price="15" qty="2" />
    </div>
  )
}
```



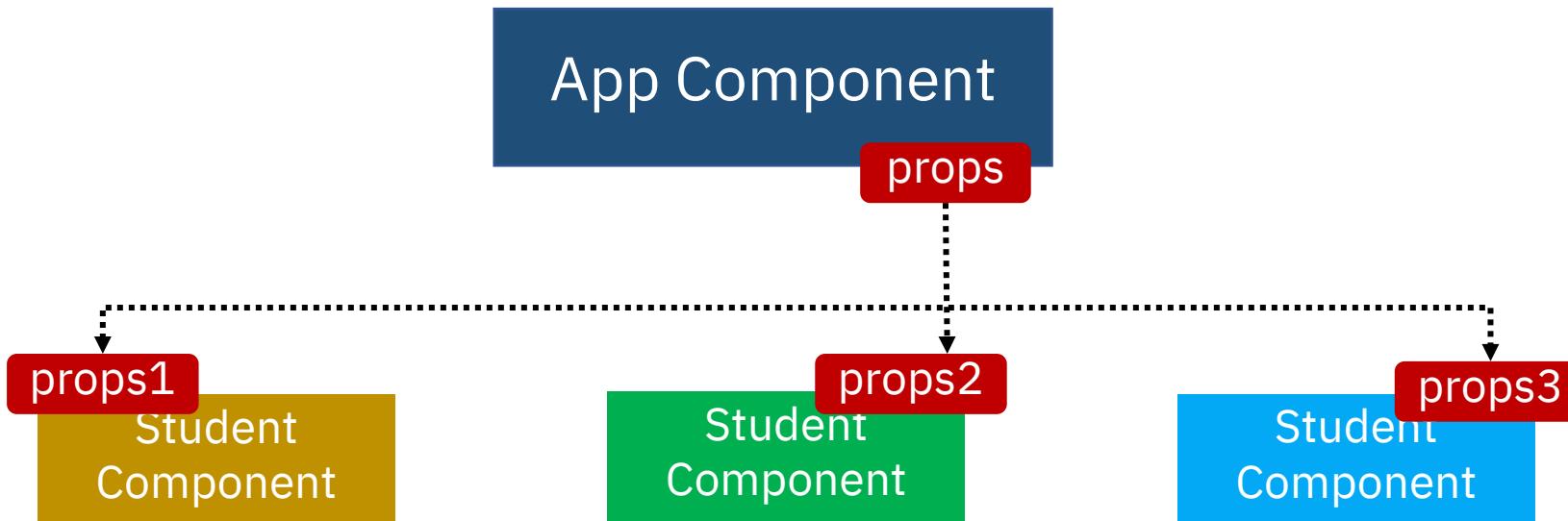
name: banana  
price: 30  
qty: 2



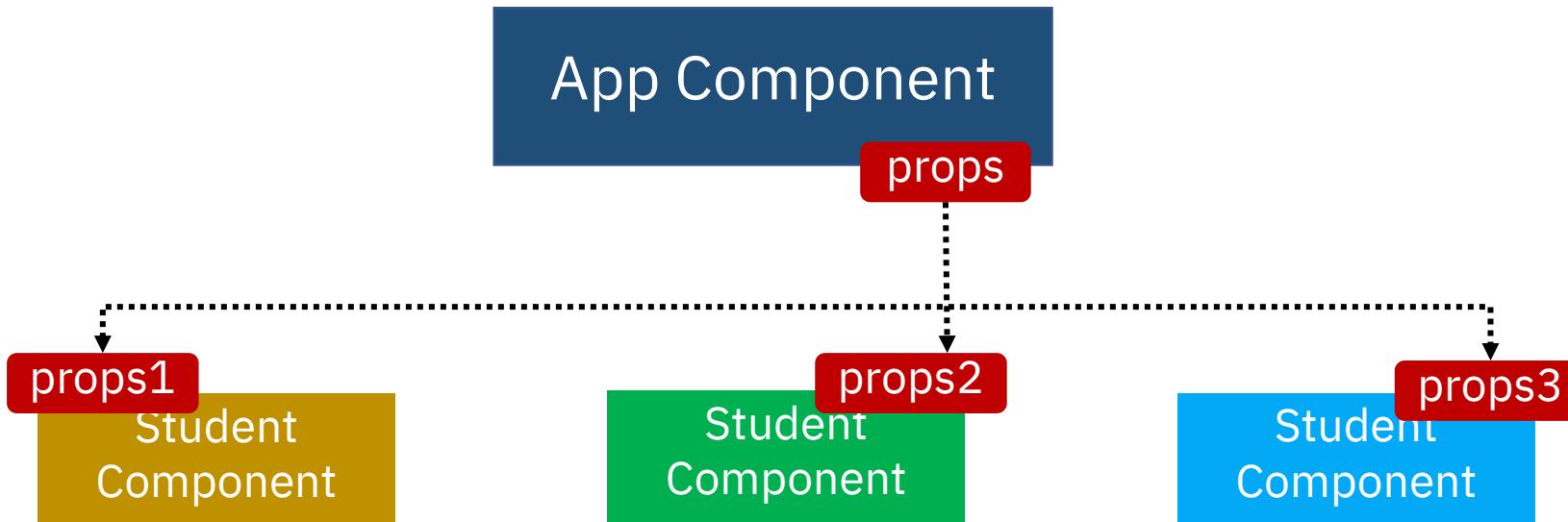
name: bread  
price: 15  
qty: 2

# รู้จักกับ props ใน React

**props** ย่อมาจาก Properties คือการส่งค่าคุณสมบัติบางอย่างจาก Parent คอมโพเนนต์ไปยัง Child คอมโพเนนต์



# การส่งผ่านค่า props จาก Parent ไปยัง Child



ทำได้ 2 รูปแบบ

- ส่งค่า props ตั้งแต่ตอนสร้าง Child component
- ส่งค่า props เพื่ออัพเดตค่าใน Child component

# วิธีผ่านค่า props ไปยัง Child component

## รูปแบบ

```
<ChildComponent propName="propsValue" />
```

- ChildComponent คือ Component ที่ได้รับ props
- propName คือ ชื่อ props ที่ต้องการส่งไปให้ child component
- propsValue คือ ค่าที่กำหนดให้ props

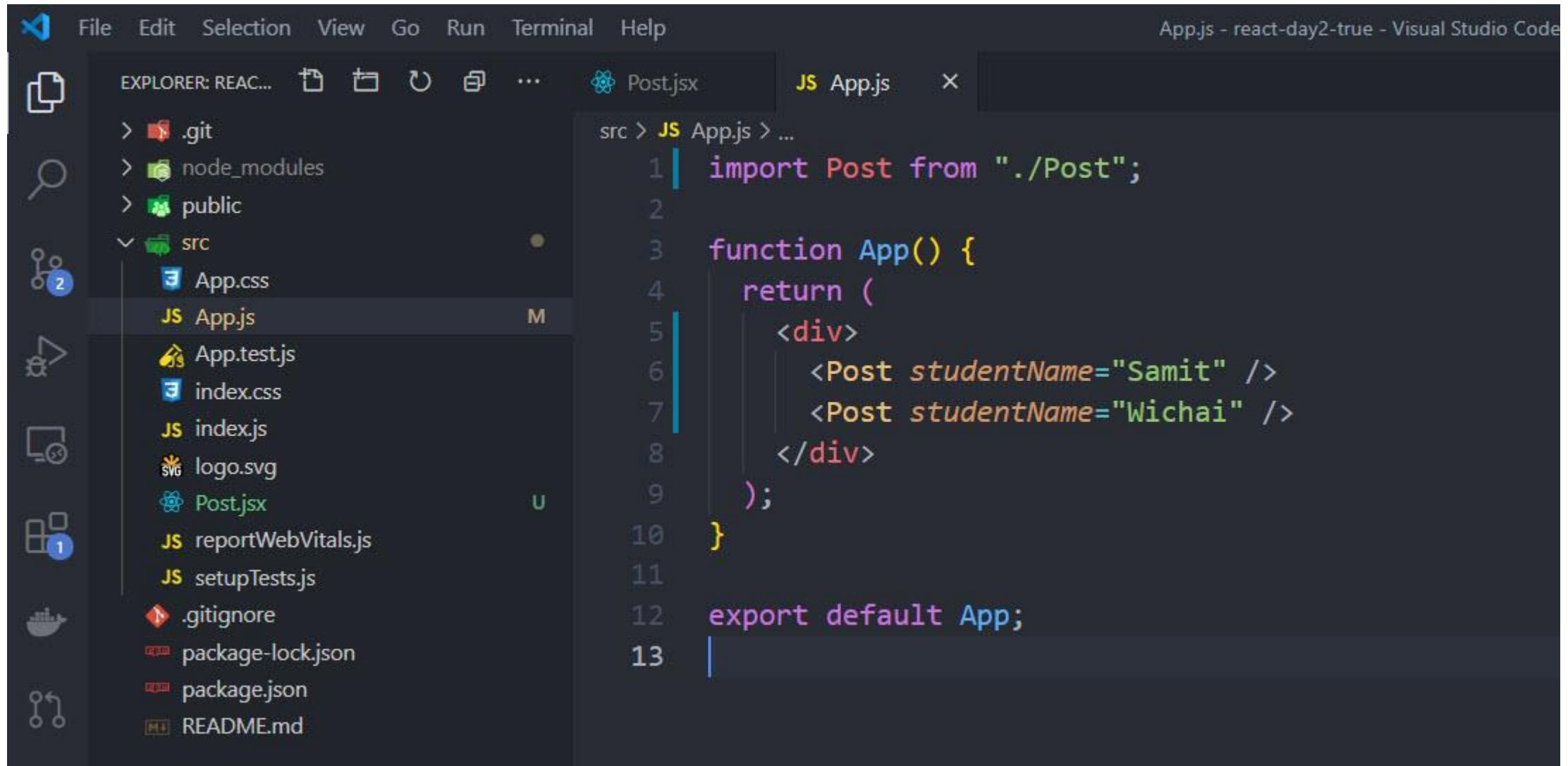
# ตัวอย่างการส่งค่า props

The screenshot shows a Visual Studio Code interface with the following details:

- File Explorer (Left):** Shows the project structure:
  - Root level: .git, node\_modules, public
  - src folder:
    - App.css, App.js, App.test.js, index.css, index.js, logo.svg
    - Post.jsx (selected)
    - reportWebVitals.js, setupTests.js
  - Others: .gitignore, package-lock.json, package.json, README.md
- Code Editor (Right):** The file `Post.jsx` contains the following code:

```
1 import React from 'react'
2
3 function Post() {
4     return (
5         <div>
6             <p>Content from Post</p>
7         </div>
8     )
9 }
10
11 export default Post
```
- Header:** Post.jsx - react-day2-true - Visual Studio Code

# ตัวอย่างการส่งค่า props

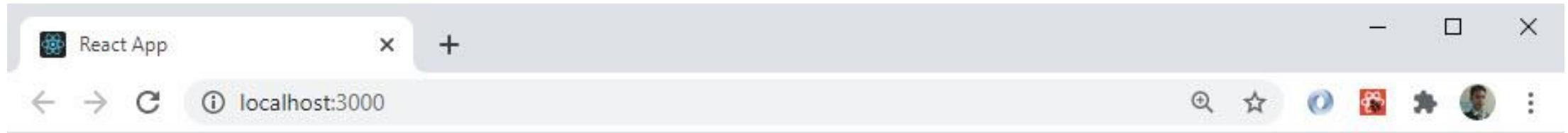


The screenshot shows the Visual Studio Code interface with the following details:

- File Menu:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Title Bar:** App.js - react-day2-true - Visual Studio Code.
- Explorer Bar (Left):** Shows the project structure:
  - REACT...
  - .git
  - node\_modules
  - public
  - src
    - App.css
    - App.js** (highlighted)
    - App.test.js
    - index.css
    - index.js
    - logo.svg
    - Post.jsx
    - reportWebVitals.js
    - setupTests.js
  - .gitignore
  - package-lock.json
  - package.json
  - README.md
- Code Editor (Right):** Displays the content of the App.js file:

```
src > JS App.js > ...
1 import Post from "./Post";
2
3 function App() {
4     return (
5         <div>
6             <Post studentName="Samit" />
7             <Post studentName="Wichai" />
8         </div>
9     );
10 }
11
12 export default App;
13 |
```

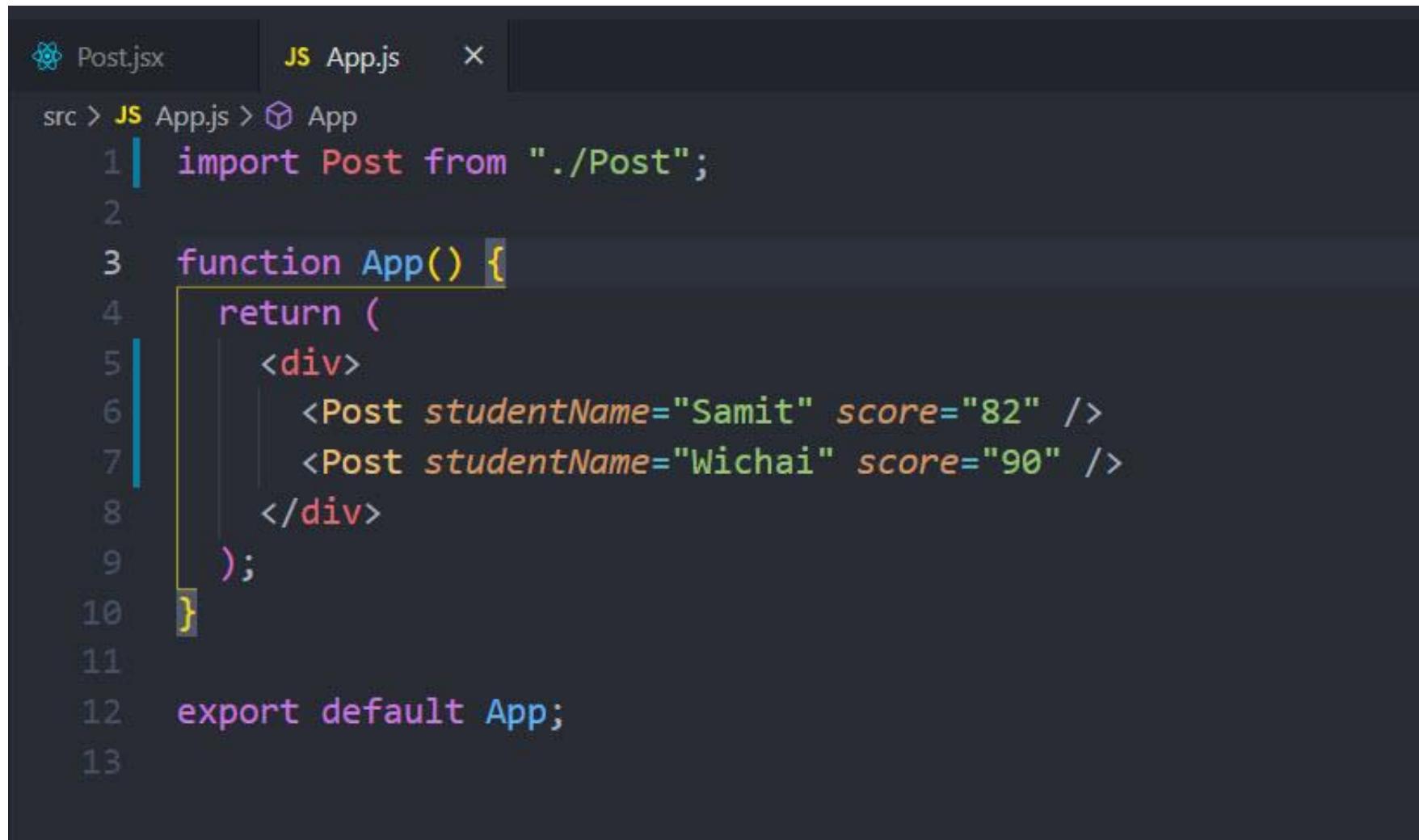
# ตัวอย่างการส่งค่า props



Content from Post

Content from Post

# การส่งค่า props ถ่ายค่าไปยัง Child component



```
Post.jsx          JS App.js      X
src > JS App.js > App
1 import Post from "./Post";
2
3 function App() {
4   return (
5     <div>
6       <Post studentName="Samit" score="82" />
7       <Post studentName="Wichai" score="90" />
8     </div>
9   );
10 }
11
12 export default App;
```

A screenshot of a code editor showing the file structure: Post.jsx, JS App.js, and App. The JS App.js file contains the following code:

```
import Post from "./Post";

function App() {
  return (
    <div>
      <Post studentName="Samit" score="82" />
      <Post studentName="Wichai" score="90" />
    </div>
  );
}

export default App;
```

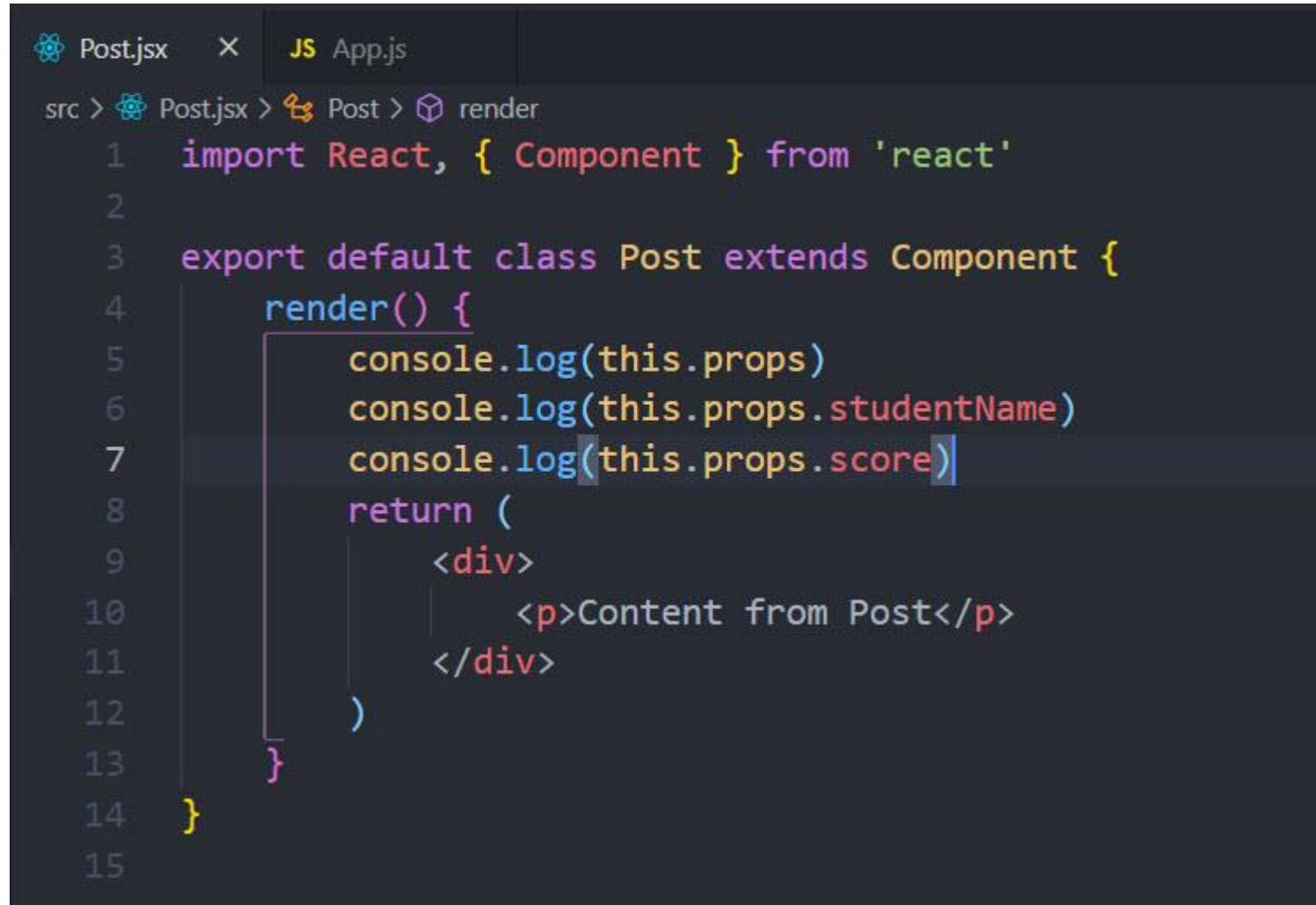
The code uses the Post component from the Post.jsx file, passing studentName and score as props. A yellow box highlights the two instances of the Post component.

# การนำค่า props มาใช้ใน component

## มี 2 แบบ คือ

- การใช้ props ในฟังก์ชัน component
- การใช้ props ในคลาส component

# ตัวอย่างการใช้งาน props ใน class component



The screenshot shows a code editor with two tabs: "Post.jsx" and "App.js". The "Post.jsx" tab is active, displaying the following code:

```
1 import React, { Component } from 'react'
2
3 export default class Post extends Component {
4   render() {
5     console.log(this.props)
6     console.log(this.props.studentName)
7     console.log(this.props.score)
8     return (
9       <div>
10         <p>Content from Post</p>
11       </div>
12     )
13   }
14 }
15 }
```

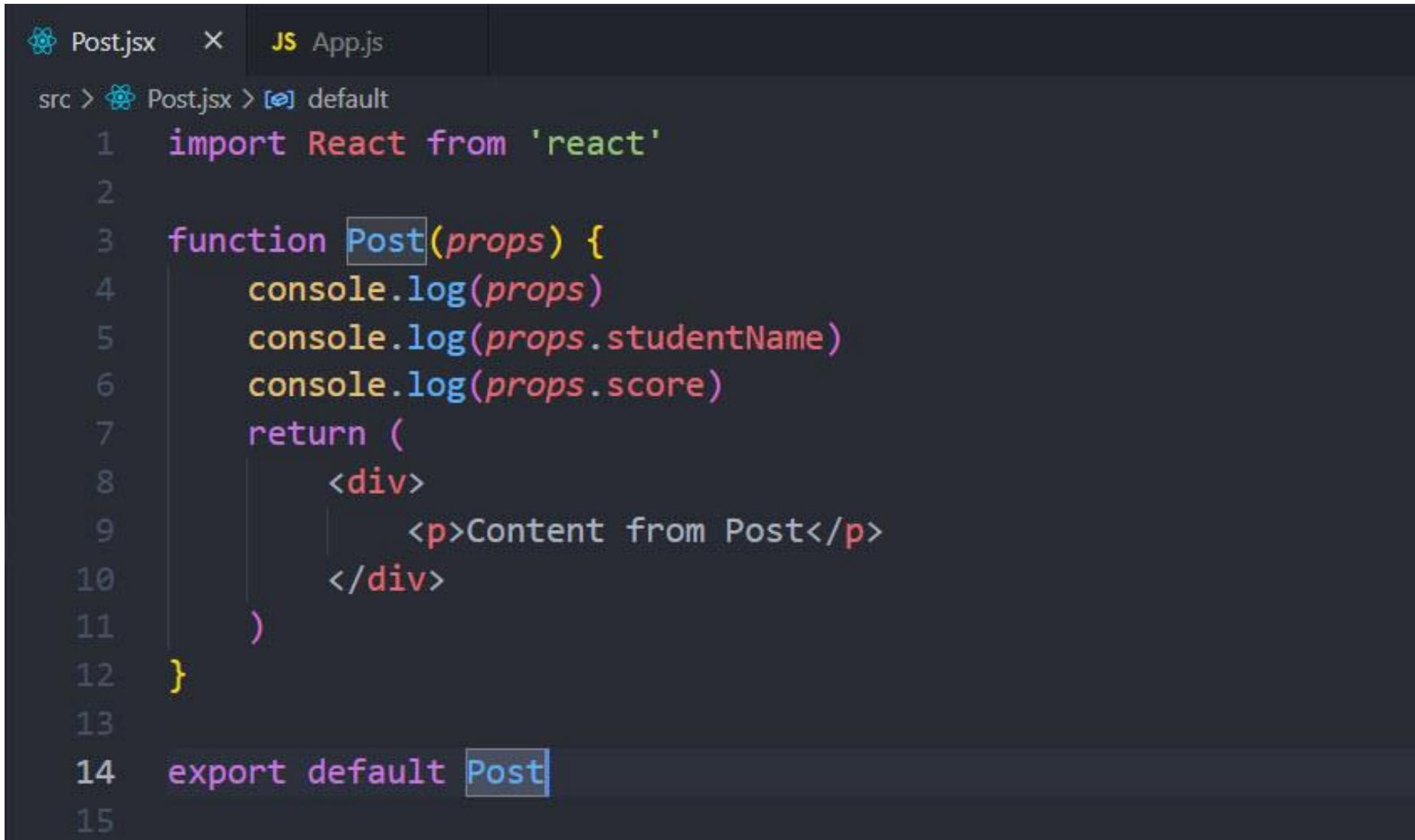
The code imports React and defines a class-based component named Post that extends Component. The render method logs the props object, its studentName, and score properties to the console, and returns a div containing a paragraph with the text "Content from Post".

# ตัวอย่างการใช้งาน props ใน class component

The screenshot shows a browser window titled "React App" at "localhost:3000". The page displays two identical components, each with the text "Content from Post". To the right, the browser's developer tools are open, specifically the "Console" tab. The console output is as follows:

```
[HMR] Waiting for update signal from log.js:24  
WDS...  
▶ {studentName: "Samit", score: "82"} Post.jsx:5  
Samit Post.jsx:6  
82 Post.jsx:7  
Post.jsx:5  
▶ {studentName: "Wichai", score: "90"}  
Wichai Post.jsx:6  
90 Post.jsx:7
```

# ตัวอย่างการใช้งาน props ใน function component

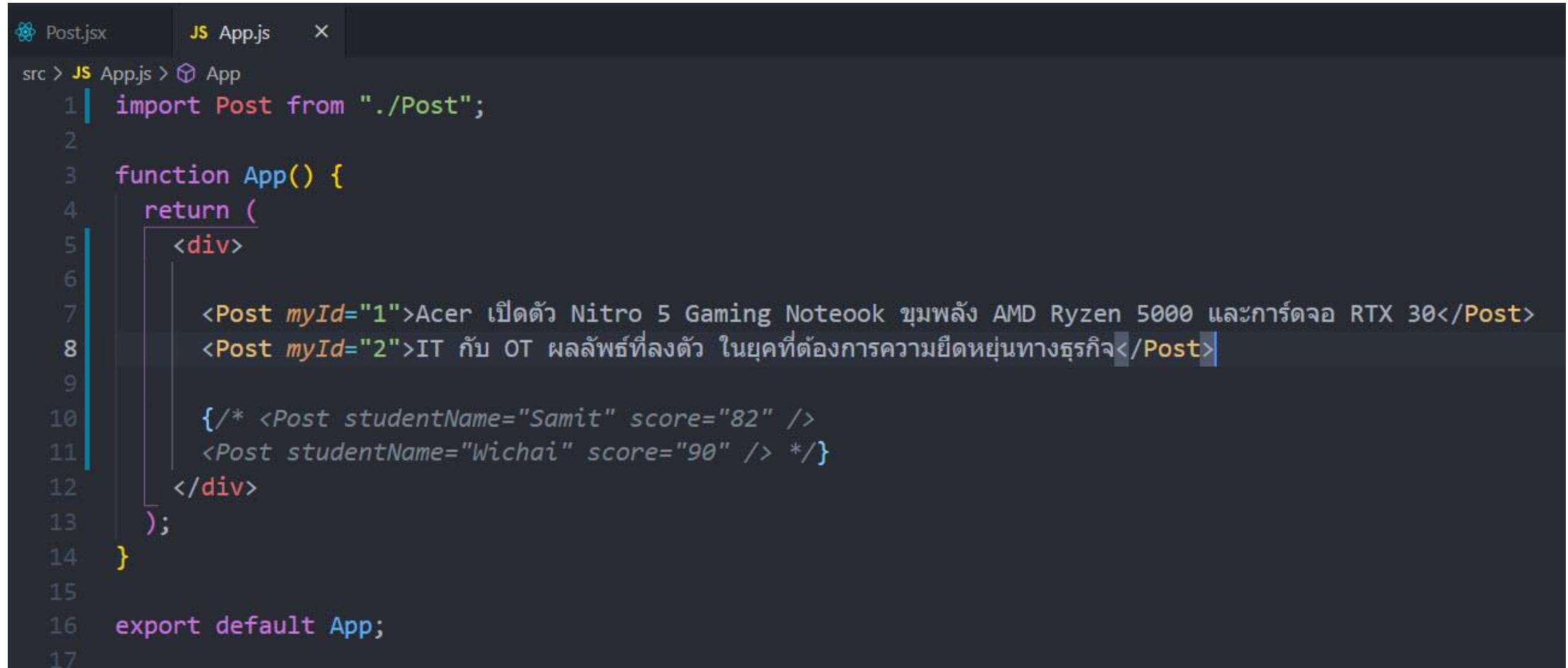


The screenshot shows a code editor with two tabs: "Post.jsx" and "App.js". The "Post.jsx" tab is active, displaying the following code:

```
1 import React from 'react'
2
3 function Post(props) {
4   console.log(props)
5   console.log(props.studentName)
6   console.log(props.score)
7   return (
8     <div>
9       <p>Content from Post</p>
10    </div>
11  )
12}
13
14 export default Post
15
```

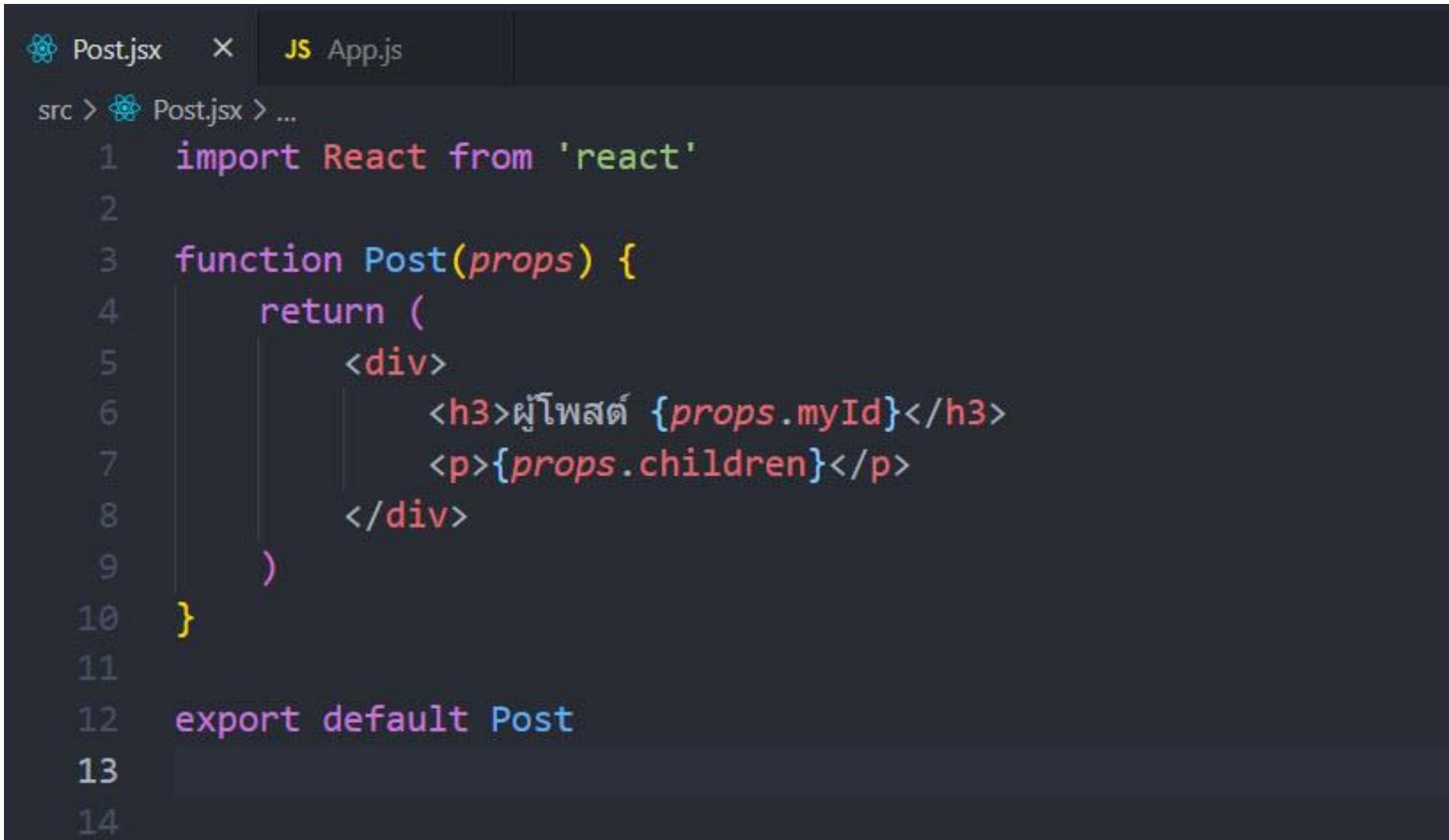
The word "props" in the first line and the parameter of the function in line 3 are highlighted with a yellow box.

# การส่งแบบมี children



```
Post.jsx      JS App.js  X
src > JS App.js > App
1 import Post from "./Post";
2
3 function App() {
4   return (
5     <div>
6
7       <Post myId="1">Acer เปิดตัว Nitro 5 Gaming Noteook ขุมพลัง AMD Ryzen 5000 และการ์ดจอ RTX 30</Post>
8       <Post myId="2">IT กับ OT ผลลัพธ์ที่ลงตัว ในยุคที่ต้องการความยืดหยุ่นทางธุรกิจ</Post>
9
10      /* <Post studentName="Samit" score="82" />
11      <Post studentName="Wichai" score="90" /> */
12
13   );
14 }
15
16 export default App;
17
```

# การส่งแบบมี children



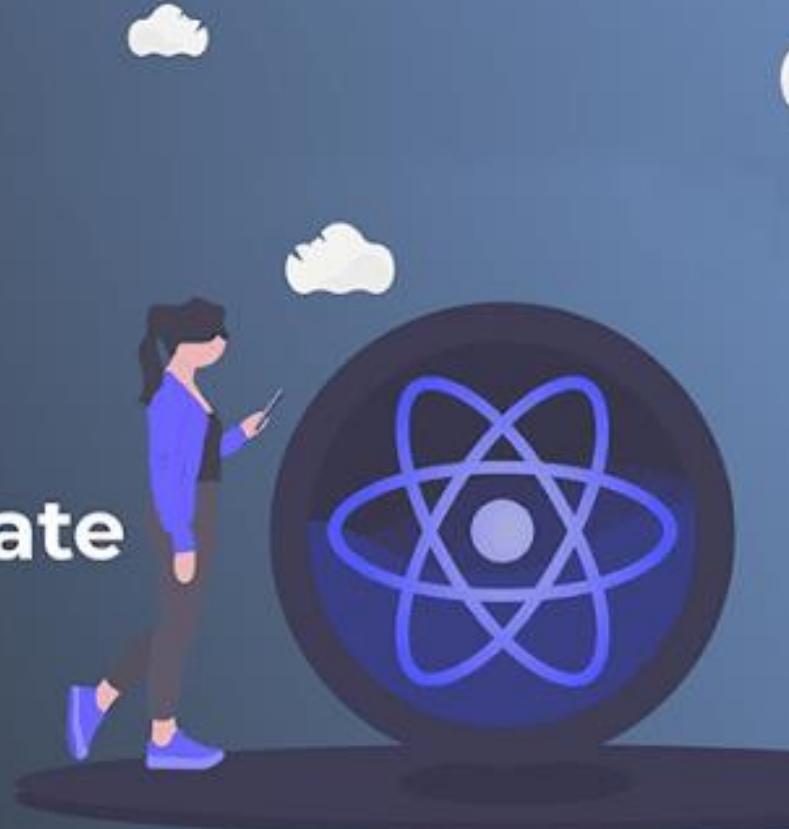
The screenshot shows a code editor with two tabs: 'Post.jsx' and 'App.js'. The 'Post.jsx' tab is active, displaying the following code:

```
1 import React from 'react'
2
3 function Post(props) {
4     return (
5         <div>
6             <h3>ผู้โพสต์ {props.myId}</h3>
7             <p>{props.children}</p>
8         </div>
9     )
10 }
11
12 export default Post
13
14
```

The code defines a functional component named 'Post' that takes a prop 'props'. Inside the component, it renders a 'div' element containing an 'h3' with the text 'ผู้โพสต์ {props.myId}' and a 'p' element with the value of 'props.children'. The 'export default Post' statement at the bottom makes the component available for import.

# React

## Manage State



# State ใน React

- รู้จัก State ใน React
- การกำหนดค่าเริ่มต้นให้กับ State
- วิธีการใช้งาน State ในคลาสและฟังก์ชัน คอมโพเนนต์
- การอัพเดตค่า state ในฟังก์ชันและคลาส คอมโพเนนต์
- การส่งค่าฟังก์ชัน (method) ไปกับ props
- การอ่านค่าพารามิเตอร์ที่ส่งไปพร้อมกับ props



# รู้จักกับ state ใน React

**State** คือ ข้อมูลที่เก็บอยู่ในคอมโพเนนต์ State ใน React เป็นคุณสมบัติที่สำคัญมาก เมื่อค่าที่อยู่ใน state เปลี่ยนแปลงไปจากเดิม React จะมีการเรนเดอร์เนื้อหาที่อยู่ในคอมโพเนนต์ใหม่

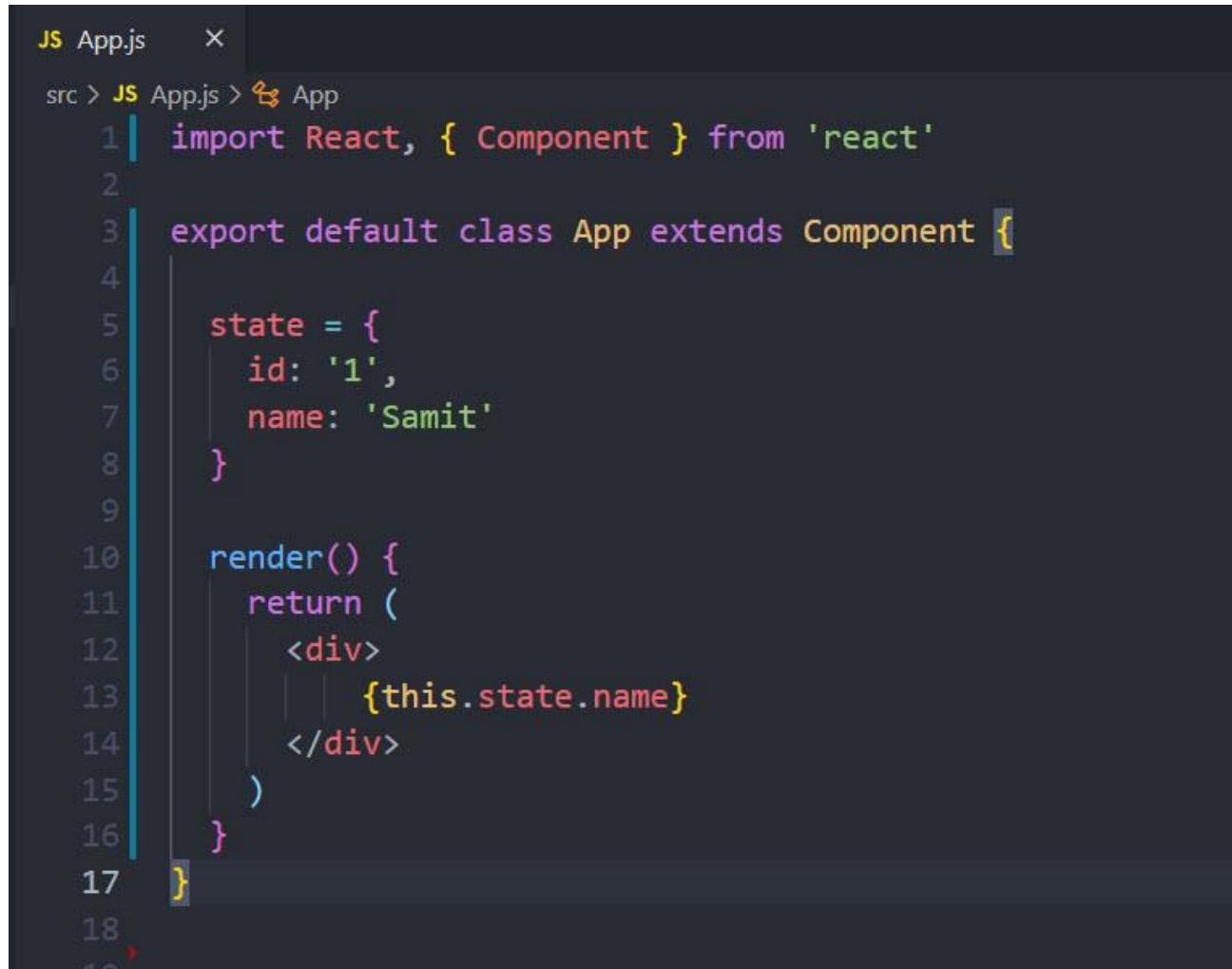
## การสร้างข้อมูลแบบ State

```
state = {  
  name: 'Samit',  
  age: 38  
}
```

# การสร้างข้อมูลแบบ State

```
state = {  
    name: 'Jeerawuth',  
    address: {  
        houseNumber: '99/1',  
        street: 'Kaew 14',  
        subDistrict : 'Klongchagpra',  
        District: 'Talingchan',  
        Province: 'Bangkok',  
        zipcode: '10170'  
    },  
    username: 'iDontKnowWhatToDo',  
    password: 'kD0F1Rz!ops1_kPdizKY',  
    hobbies: ['Tennis', 'GoIf', 'Swimming']  
}
```

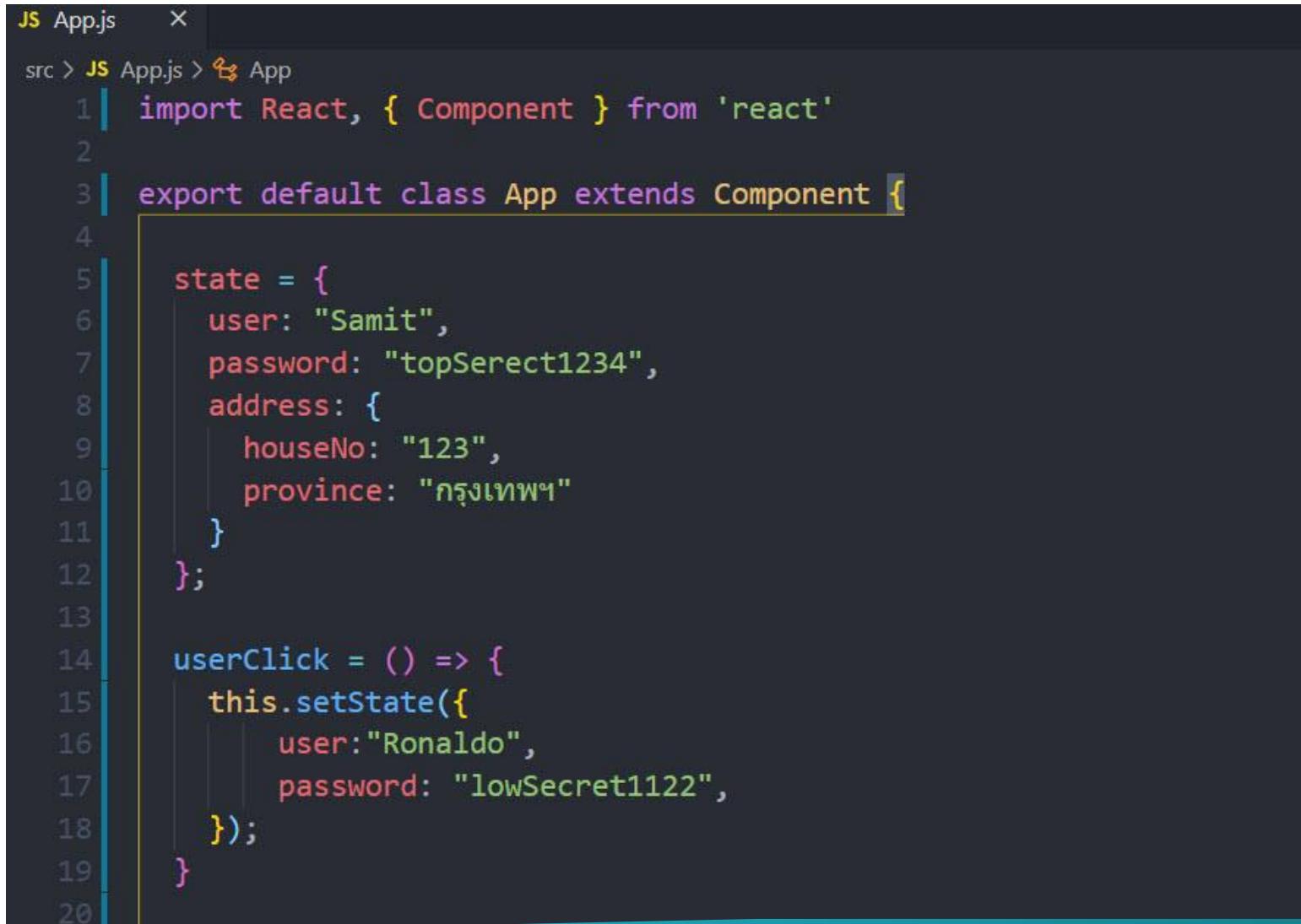
# state ใน class component



The screenshot shows a code editor with a dark theme. The file is named 'App.js' and is located in a 'src' directory. The code defines a class component 'App' that extends 'Component' from 'react'. It has a state object with 'id' and 'name' properties. The 'render' method returns a `<div>` element containing the value of 'this.state.name'. The code is numbered from 1 to 18.

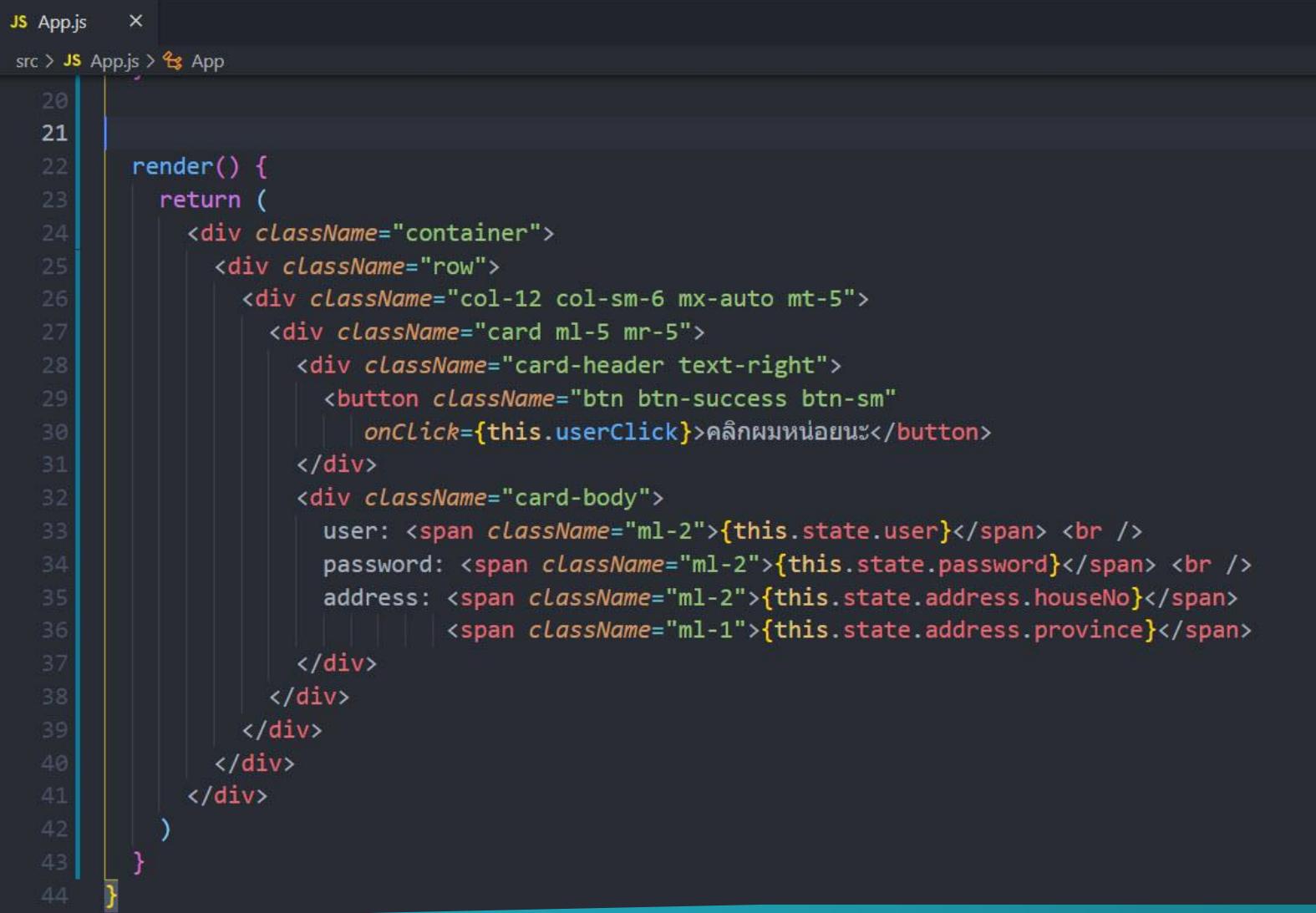
```
JS App.js    X
src > JS App.js > App
1 | import React, { Component } from 'react'
2 |
3 | export default class App extends Component {
4 |
5 |   state = {
6 |     id: '1',
7 |     name: 'Samit'
8 |   }
9 |
10 |   render() {
11 |     return (
12 |       <div>
13 |         {this.state.name}
14 |       </div>
15 |     )
16 |   }
17 |
18 | }
```

# ตัวอย่างการอัพเดตค่า state ด้วย setState()



```
JS App.js X
src > JS App.js > App
1 import React, { Component } from 'react'
2
3 export default class App extends Component {
4
5   state = {
6     user: "Samit",
7     password: "topSerect1234",
8     address: {
9       houseNo: "123",
10      province: "กรุงเทพฯ"
11    }
12  };
13
14  userClick = () => {
15    this.setState({
16      user: "Ronaldo",
17      password: "lowSecret1122",
18    });
19  }
20}
```

# ตัวอย่างการอัพเดทค่า state ด้วย setState()

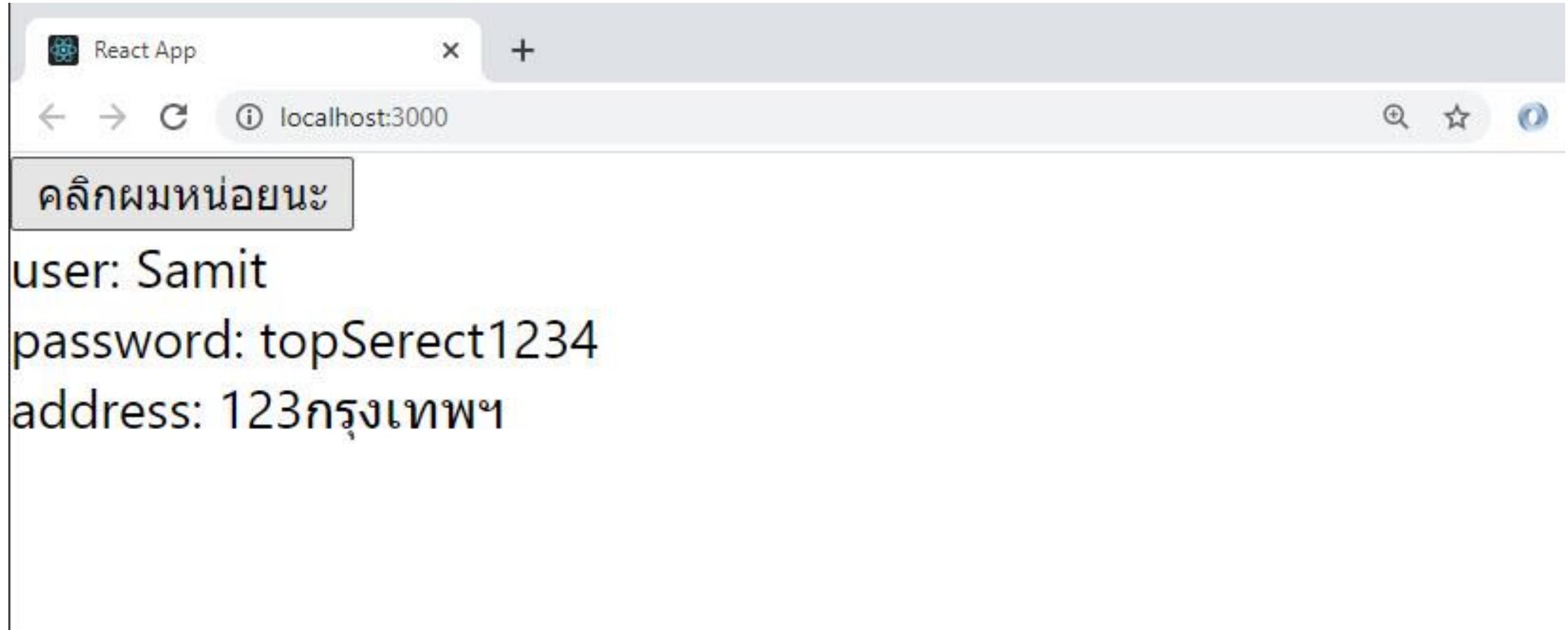


The screenshot shows a code editor window with a dark theme. The file is named `App.js`. The code is a React component with the following structure:

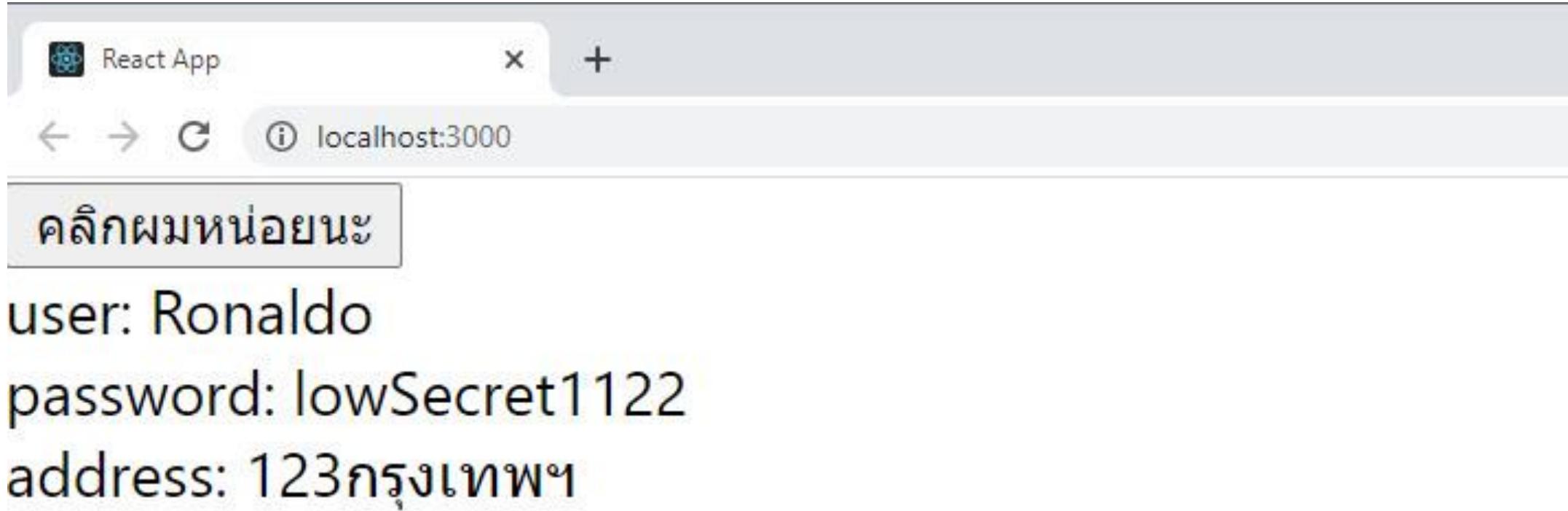
```
JS App.js
src > JS App.js > App
20
21
22 render() {
23   return (
24     <div className="container">
25       <div className="row">
26         <div className="col-12 col-sm-6 mx-auto mt-5">
27           <div className="card ml-5 mr-5">
28             <div className="card-header text-right">
29               <button className="btn btn-success btn-sm"
30                 onClick={this.userClick}>คลิกผมหน่อยนะ</button>
31             </div>
32             <div className="card-body">
33               user: <span className="ml-2">{this.state.user}</span> <br />
34               password: <span className="ml-2">{this.state.password}</span> <br />
35               address: <span className="ml-2">{this.state.address.houseNo}</span>
36                 <span className="ml-1">{this.state.address.province}</span>
37               </div>
38             </div>
39           </div>
40         </div>
41       </div>
42     )
43   }
44 }
```

The code uses `onClick` to trigger a function named `userClick` when a button is clicked. The component also displays user information using `{this.state.user}`, `{this.state.password}`, and `{this.state.address}`.

# ตัวอย่างการอัพเดตค่า state ด้วย setState()



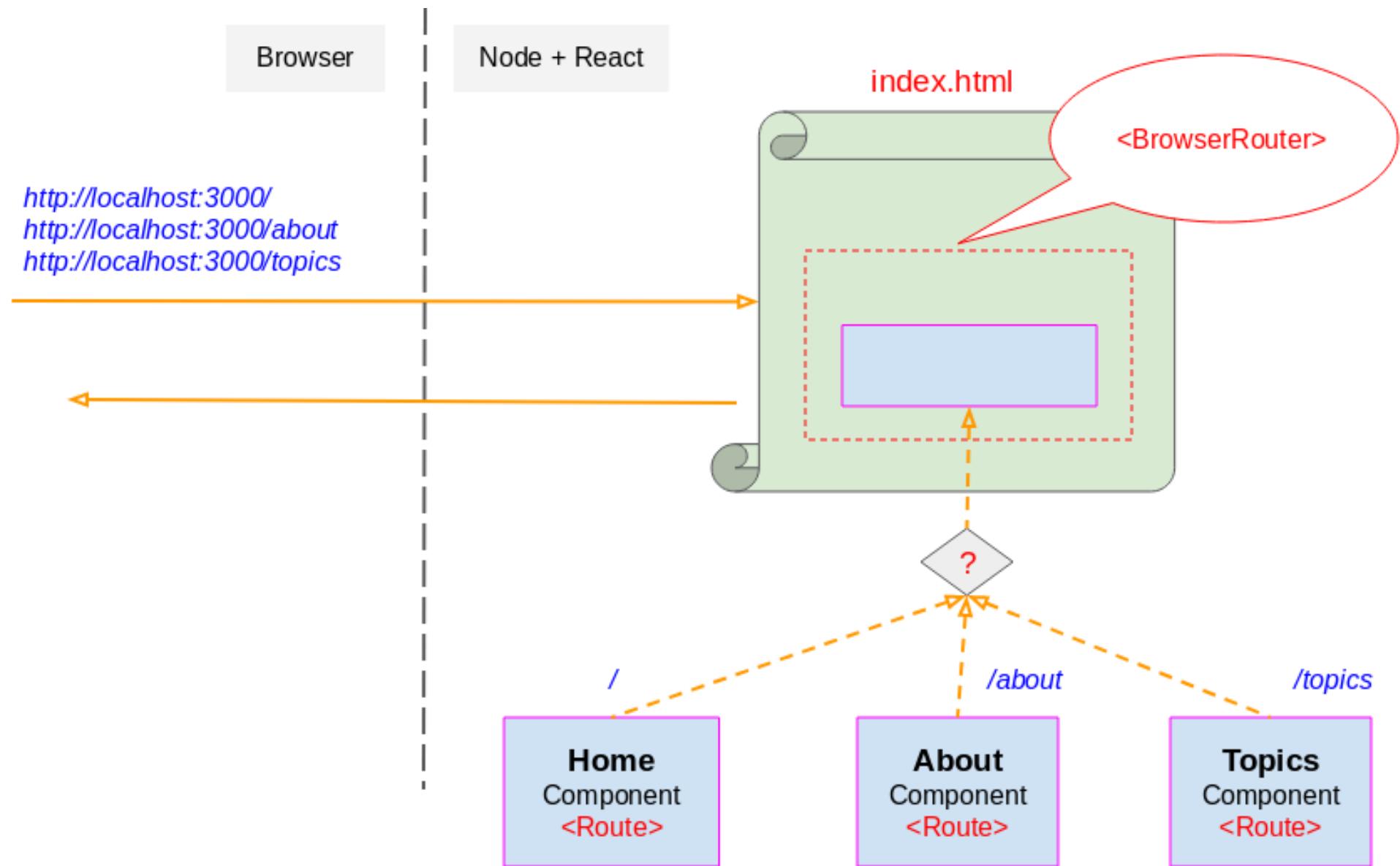
# ตัวอย่างการอัพเดตค่า state ด้วย setState()





# การใช้งาน React Router

- รู้จัก React Router
- การติดตั้ง React Router
- แนะนำภาพรวมของการใช้งาน React Router
- การกำหนด url ที่จะต้องใช้ในแอพพลิเคชัน
- การกำหนด Link
- การแสดงสถานะปัจจุบันของลิงก์
- การส่งค่าไปพร้อมกับลิงก์บน url
- การส่งค่าคิวเรสตริงไปพร้อม url
- การใช้งาน switch
- การใช้งาน redirect
- การกำหนดเงื่อนไขในการเปลี่ยนเส้นทาง
- การเขียนและแสดงข้อผิดพลาด 404 เมื่อไม่พบ url ที่ร้องขอ

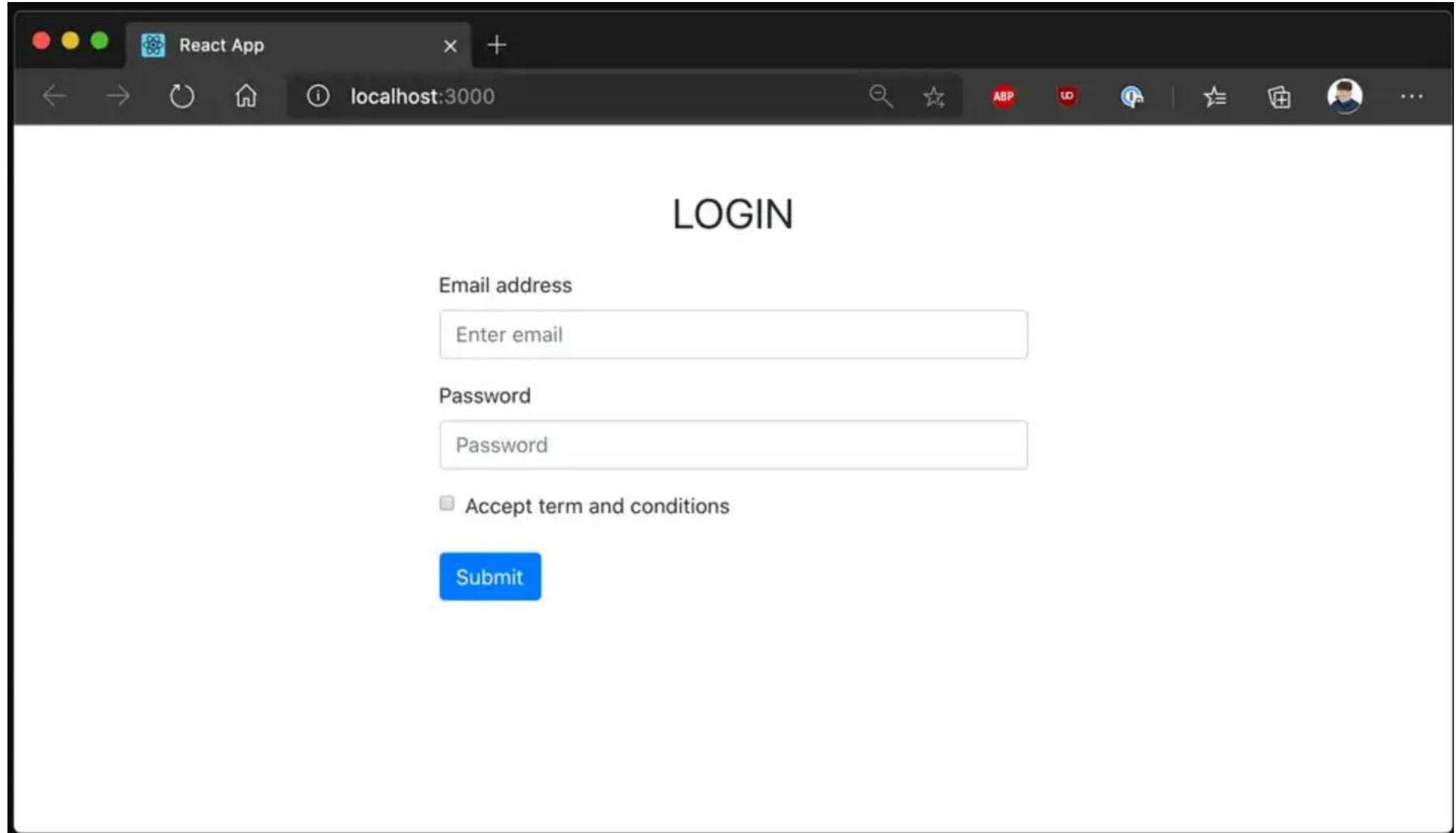




# React Hook Form

## การทำงานกับแบบฟอร์มใน React

- สร้างแบบฟอร์มใน jsx ด้วย bootstrap 5
- การสร้างแบบฟอร์มด้วยวิธีอ้างอิงกับ state
- การใช้ React Hook form จัดการฟอร์มอย่างมืออาชีพ
- การตรวจสอบความถูกต้องของแบบฟอร์ม
- การรับค่าจากฟอร์มไปใช้งาน
- ตัวอย่างการเขียนแบบฟอร์ม ล็อกอินและลงทะเบียน



# React and NodeJS with Docker Workshop Progress



Day 1



Day 2



Day 3



Day 4



Day 5

# DAY 3

**Section 6:** สร้าง Workshop Rest API ด้วย Node.JS  
และ Express.JS



# รู้จัก Node.JS พื้นฐานการทำงานกับ Node.JS



# What is Node.js ?

Modules

Built-in Modules

Package Manager

Web Server



# What is Node.js ?

Modules

Built-in Modules

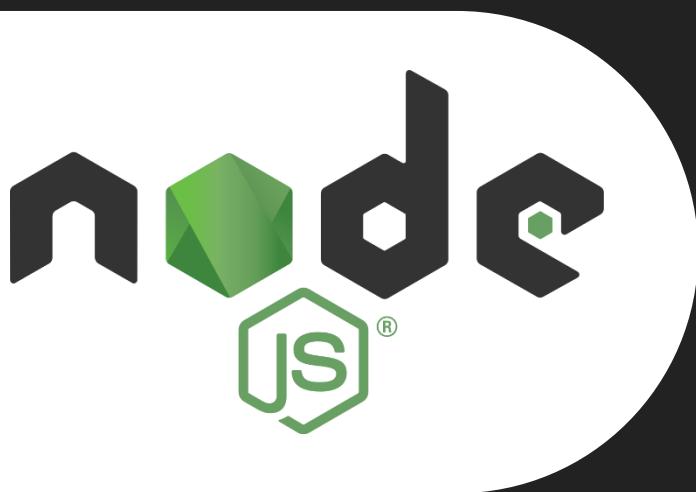
Package Manager

Web Server



# What is Node.js

“JavaScript Runtime Environment”



V8

Node.js ไม่ใช่ตัวภาษา JavaScript แต่เป็นส่วนของแวดล้อมที่เตรียมไว้ใช้กับ JavaScript บนแพลตฟอร์มทำงาน (JavaScript Runtime Environment) ซึ่ง Node.js ช่วยให้เราเขียน JavaScript โดยไม่จำเป็นต้องทำงานบนブラウเซอร์ และยังสามารถเขียนโค้ด JavaScript เพื่ออ่านเขียนไฟล์ หรือติดต่อฮาร์ดแวร์ที่อยู่ในเครื่องคอมพิวเตอร์ได้

Node.js ประกอบด้วย JavaScript Engine V8 ของ Google Chrome ใช้เพื่อตีความคำสั่ง JavaScript และใน Node.js ยังมีโค้ดภาษา C++ ครอบ V8 ไว้อีกด้วย

Node.js มีลักษณะการทำงานเป็นแบบ Non Blocking IO ซึ่งหมายถึงแอ�� พลิเคชันสามารถทำงานได้ต่อเนื่อง โดยไม่ต้องรอผลลัพธ์บางอย่างที่ต้องใช้เวลา จึงทำงานได้รวดเร็วและมีประสิทธิภาพมาก

# JavaScript Engine



Chakra



V8

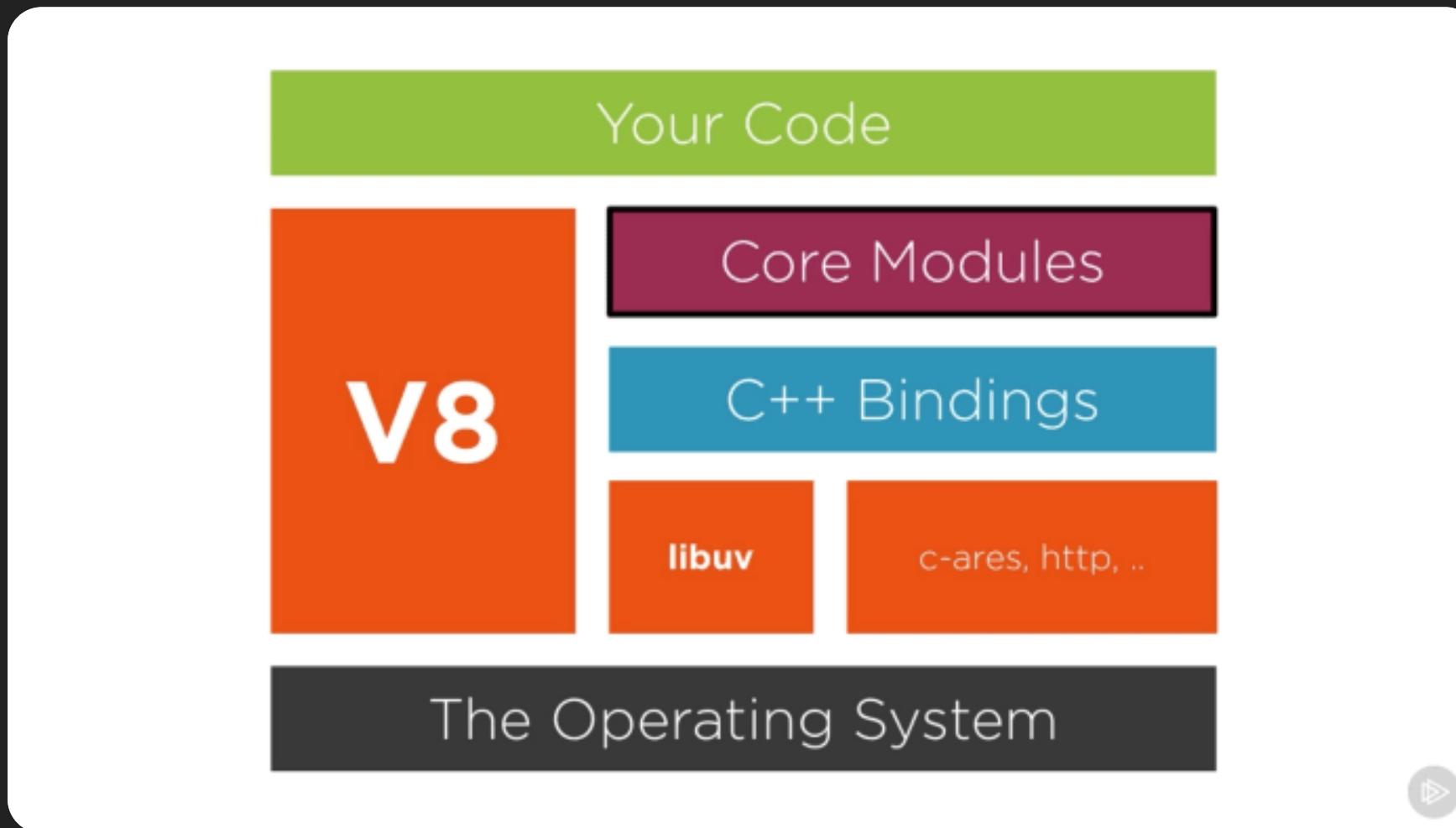


SpiderMonkey

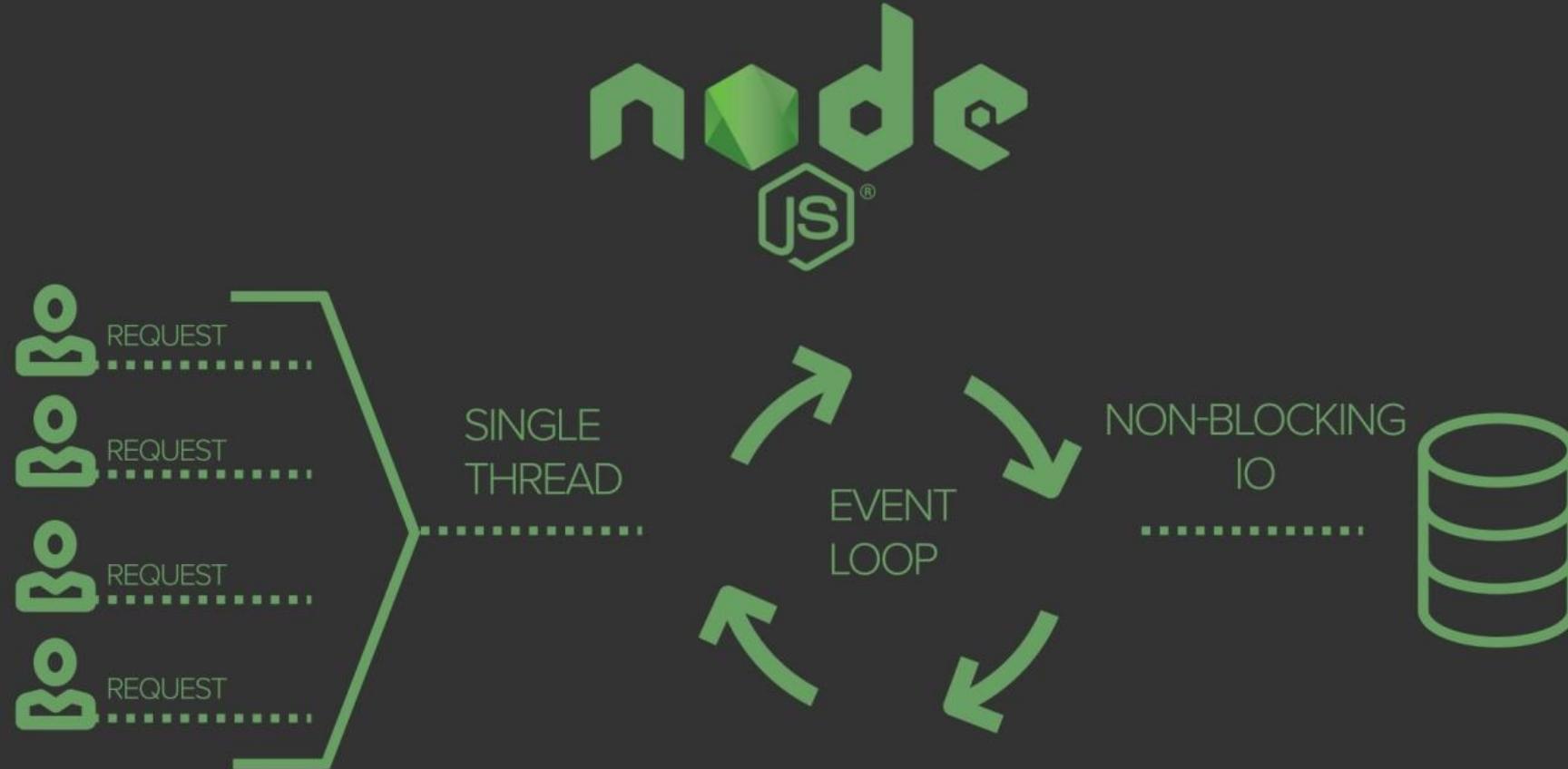


JavaScriptCore

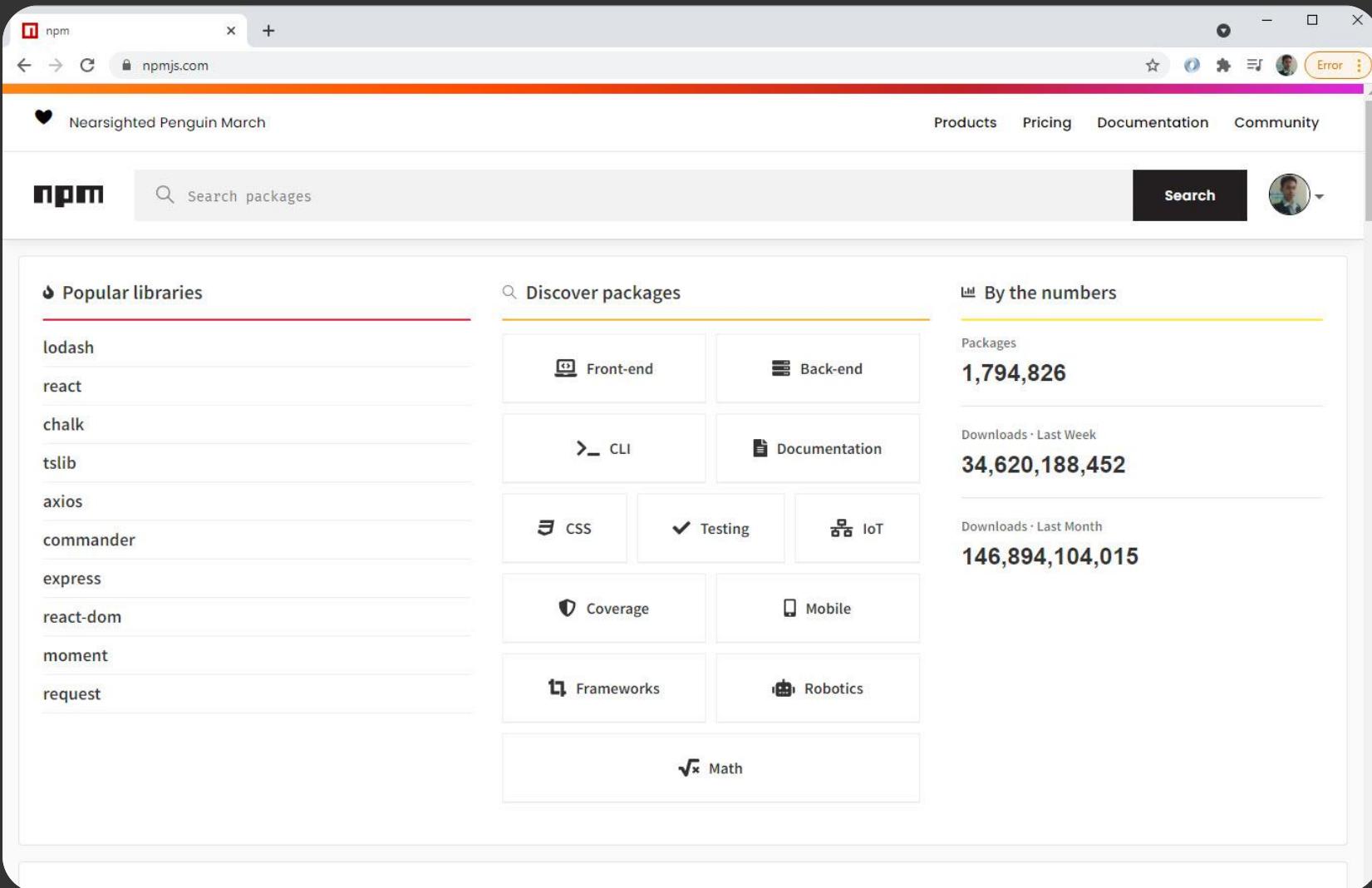
# What is Node.js



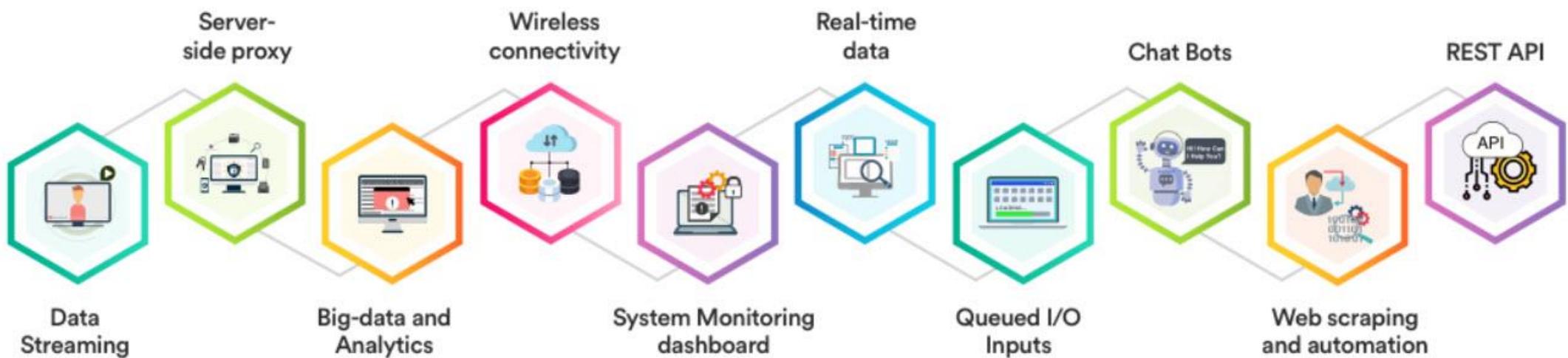
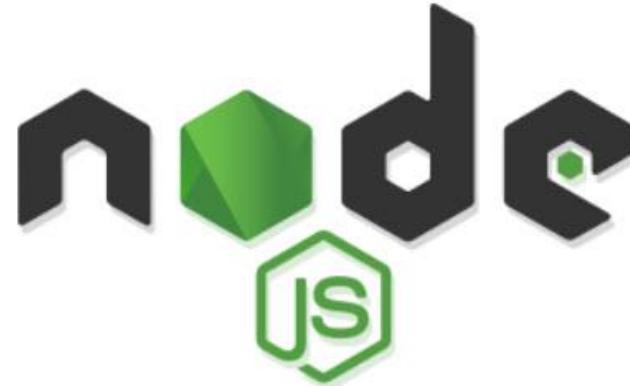
# Blocking I/O vs Non-Blocking I/O



# Node Package Manager



# Node.js Use Cases



# Top company use Node.js

**Node JS Applications**

Walmart 

ebay 

YAHOO! 

PayPal 

NETFLIX 

NASA 

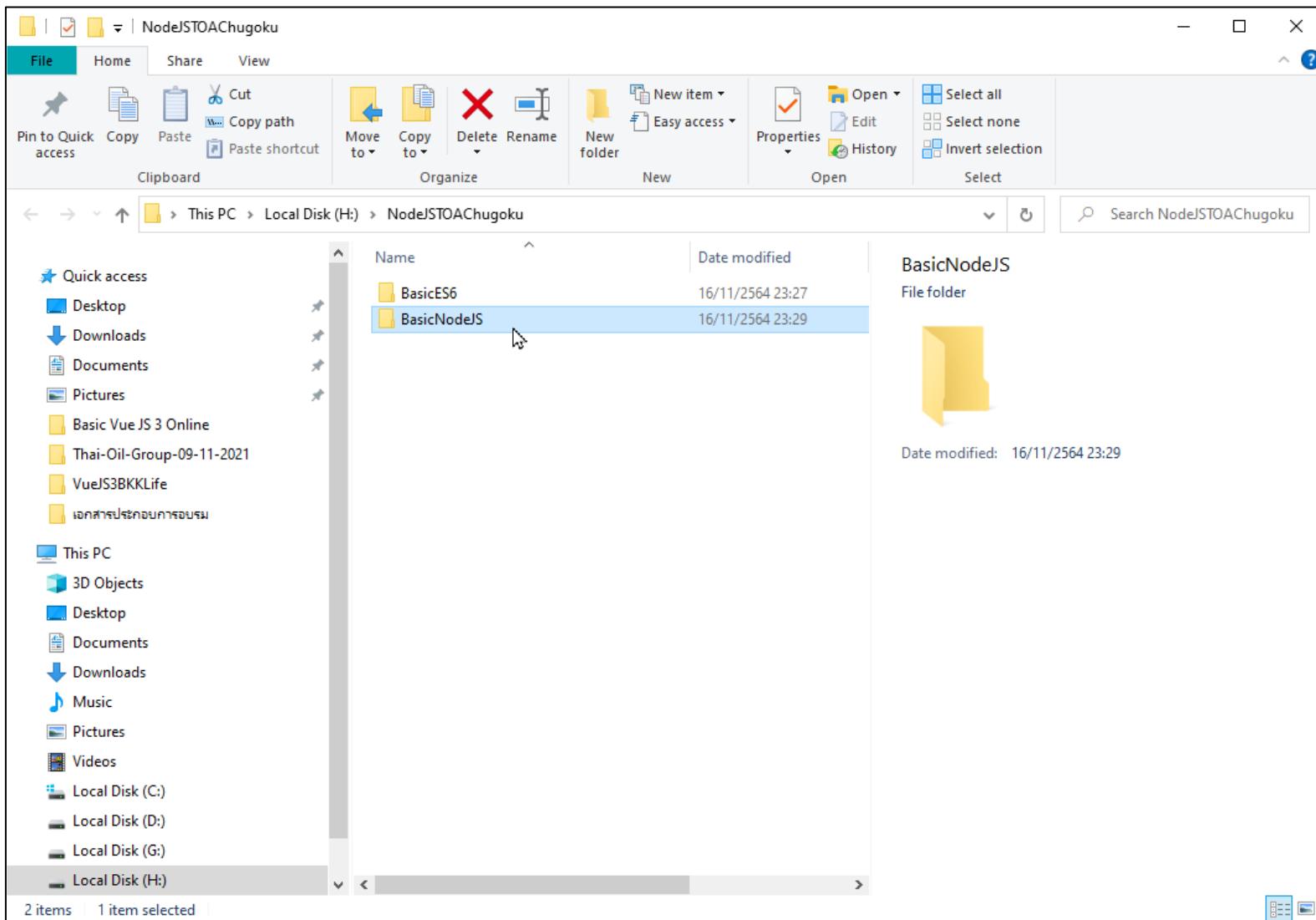
Trello 

GROUPON 

UBER 

LinkedIn 

# Create project folder “NodeJSUTAC/BasicNodJS”



# Exercise 1: What is node JS

The screenshot shows a Visual Studio Code interface with the following details:

- File Explorer:** Shows a project structure under "NODEJSTOACHUGOKU" with folders "BasicES6", "BasicNodeJS", and "1\_what\_is\_node\_js". Inside "1\_what\_is\_node\_js", there is an "index.js" file.
- Code Editor:** Displays the content of "index.js":

```
1 console.log("Hello World!");
2 console.log(5+8);
3 const result = 5 * 3;
4 console.log(result);
```
- Terminal:** Shows the output of running the script:

```
H:\NodeJSTOACHugoku\BasicNodeJS\1_what_is_node_js>node index.js
Hello World!
15
```
- Status Bar:** Shows file statistics: Ln 5, Col 1, Spaces: 4, UTF-8, CRLF, {} JavaScript, 88, ✓ Prettier.

What is Node.js ?

Modules

Built-in Modules

Package Manager

Web Server



# แนะนำ Module ใน Node.js



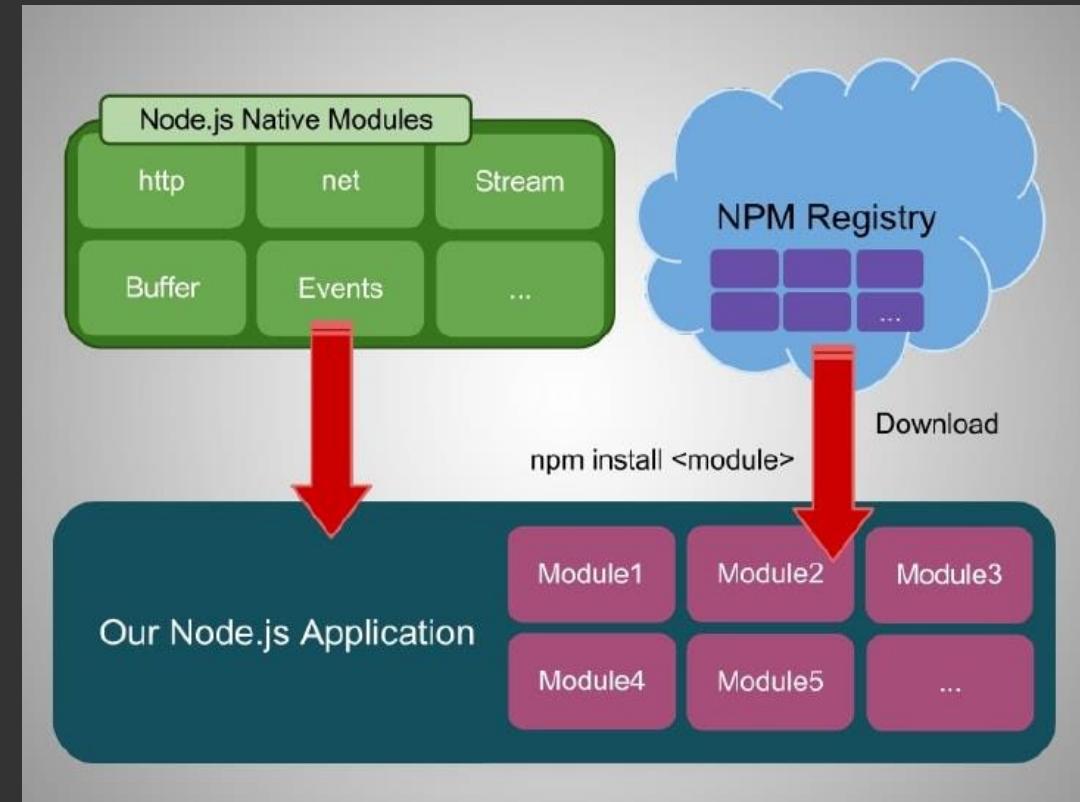
- การใช้งานโมดูลเบื้องต้น
- โกลบล้อบเจ็กต์ (Global Object) ใน Node.js
- รู้จักและใช้งาน npm เช่นการสร้างไฟล์ package.json การติดตั้ง module
- ขอบเขตของตัวแปรในแต่ละ module
- การสร้าง Module ใหม่
- การ exports module ในแบบ object
- การ exports พิงก์ชัน คลาส ค่าคงที่ และข้อมูลหลายรูปแบบจาก Module

# What is modules ?

โมดูล (Module) คือ แพ็คเกจ หรือไลบรารีที่อยู่ใน Node.js โดยแต่ละโมดูลจะเป็นอิสระจากโมดูลอื่นๆ หากแอพพลิเคชันต้องการใช้โมดูลใด ก็ให้โหลดโมดูลที่ต้องการมา自己ปั๊งจุบัน

เมื่อติดตั้ง Node.js ลงในเครื่องของเรา จะได้โมดูลติดมาพร้อมกับ Node.js จำนวนหนึ่ง เราเรียกว่า Build-in Modules

นอกจากนี้เรายังสามารถสร้างโมดูลเองหรือดาวน์โหลดเพิ่มเติมจาก NPM Registry ของ Node.js ได้เพิ่มเติมอีกด้วย



# Exercise 2: Modules (Export method 1)

The screenshot shows a VS Code interface with two code editors and a terminal.

**utils.js:**

```
// Export แบบที่ 1
exports.calculateVat = function calculateVat(money, vat){
    return money * vat / 100;
}

exports.sayHello = function sayHello(){
    console.log("Hello");
}
```

**index.js:**

```
const utils = require('./utils'); // ปกติ

utils.sayHello();
const vat = utils.calculateVat(100, 7);
console.log(vat);
```

**Terminal Output:**

```
H:\NodeJSTOAChugoku\BasicNodeJS\2_modules>node index.js
Hello
7
```

# Exercise 2: Modules (Export method 2)

The screenshot shows a VS Code interface with two code editors and a terminal.

**utils.js:**

```
// Export แบบที่ 2
function calculateVat(money, vat){
    return money * vat / 100;
}

function sayHello(){
    console.log("Hello");
}

module.exports = {
    calculateVat,
    sayHello
}
```

**index.js:**

```
const utils = require('./utils'); // ปกติ
utils.sayHello();
const vat = utils.calculateVat(100, 7);
console.log(vat);
```

**Terminal Output:**

```
H:\NodeJSTOAChugoku\BasicNodeJS\2_modules>node index.js
Hello
7
```

# Exercise 2: Modules Import with *Destructuring*

The screenshot shows a VS Code interface with two code files and a terminal window.

**utils.js:**

```
// Export แบบที่ 2
function calculateVat(money, vat){
    return money * vat / 100;
}

function sayHello(){
    console.log("Hello");
}

module.exports = {
    calculateVat,
    sayHello
}
```

**index.js:**

```
// const utils = require('./utils'); // ปกติ
// Destructuring
const { sayHello, calculateVat } = require('./utils');

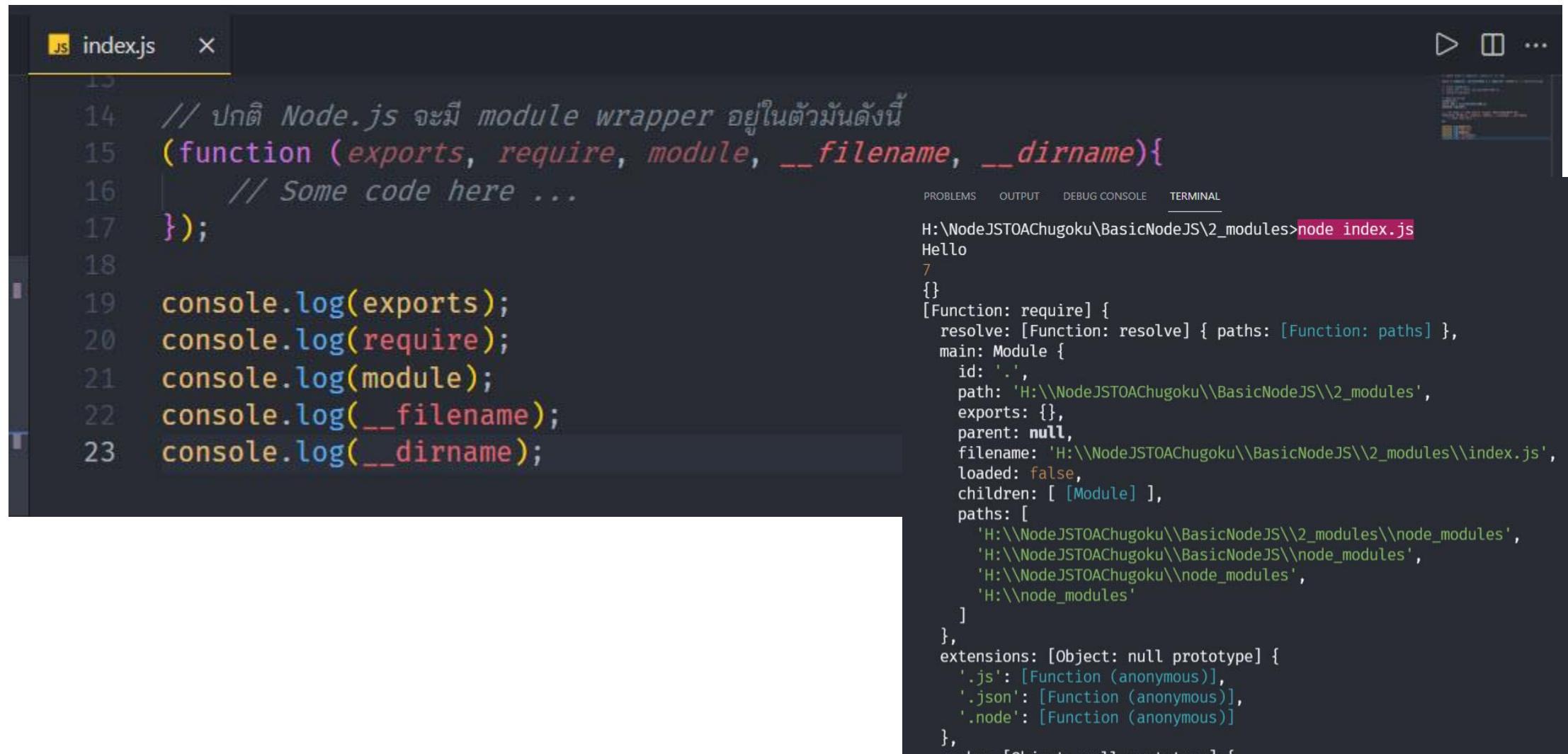
// utils.sayHello();
// const vat = utils.calculateVat(100,7);
// console.log(vat);

// Destructuring
sayHello();
const vat = calculateVat(100,7);
console.log(vat);
```

**Terminal Output:**

```
H:\NodeJSTOACHugoku\BasicNodeJS\2_modules>node index.js
Hello
7
```

## Exercise 2: Module wrapper



The screenshot shows a VS Code interface with the following details:

- File:** index.js
- Code Content:**

```
14 // ปกติ Node.js จะมี module wrapper อยู่ในตัวมันดังนี้
15 (function (exports, require, module, __filename, __dirname){
16     // Some code here ...
17 });
18
19 console.log(exports);
20 console.log(require);
21 console.log(module);
22 console.log(__filename);
23 console.log(__dirname);
```
- Terminal Output:**

```
H:\NodeJSTOChugoku\BasicNodeJS\2_modules>node index.js
Hello
7
{}
[Function: require] {
  resolve: [Function: resolve] { paths: [Function: paths] },
  main: Module {
    id: '.',
    path: 'H:\\NodeJSTOChugoku\\BasicNodeJS\\2_modules',
    exports: {},
    parent: null,
    filename: 'H:\\NodeJSTOChugoku\\BasicNodeJS\\2_modules\\index.js',
    loaded: false,
    children: [ [Module] ],
    paths: [
      'H:\\NodeJSTOChugoku\\BasicNodeJS\\2_modules\\node_modules',
      'H:\\NodeJSTOChugoku\\BasicNodeJS\\node_modules',
      'H:\\NodeJSTOChugoku\\node_modules',
      'H:\\node_modules'
    ],
    extensions: [Object: null prototype] {
      '.js': [Function (anonymous)],
      '.json': [Function (anonymous)],
      '.node': [Function (anonymous)]
    },
    cache: [Object: null prototype] {
```

What is Node.js ?

Modules

Built-in Modules

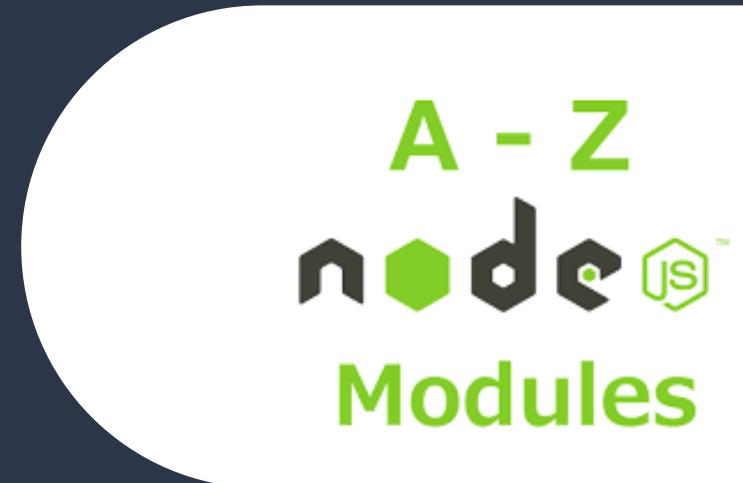
Package Manager

Web Server

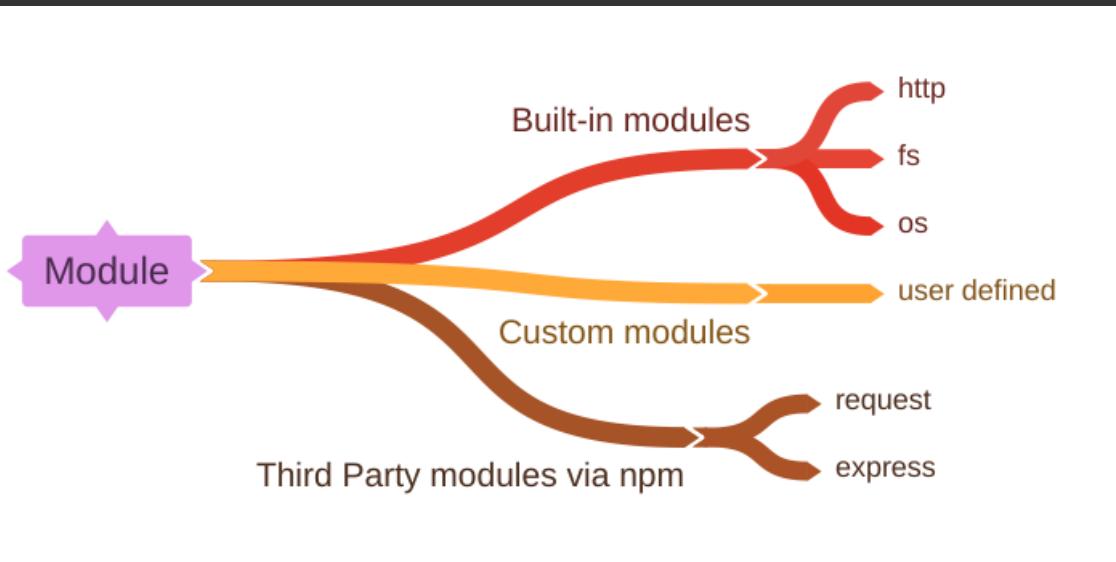


# ตัวอย่าง Module กี่มาร์ต์กับ Node.js

- รู้จัก module Path
- การใช้ module OS
- การจัดการไฟล์ด้วย module File System
- การใช้งาน module Event
- การผ่านค่าพารามิเตอร์เข้าไปใน event
- การใช้งาน HTTP Module ในเบื้องต้น
- วิธีการใช้งาน HTTP Module จัดการ http request
- การกำหนดเงื่อนไขใน http response



# Built-in Modules



ใน Node.js ได้เตรียมโมดูลไว้ให้เราใช้งานอยู่ส่วนหนึ่ง เช่น

1. File System (`fs`)
2. Path (`path`)
3. Operating System (`os`)
4. Events (`events`)
5. HTTP (`http`)

โดยแต่ละโมดูลจะมีหลักการพื้นฐานเดียวกัน คือ เริ่มจากโหลดโมดูลที่ต้องการมาเก็บยังตัวแปรที่เป็นค่าคงที่ จากนั้นจึงนำค่าพร้อมเพอตีหรือเมธอดที่อยู่ในตัวแปรไปใช้งานต่อไป

# Exercise 3: Built-in Module (Path)

The screenshot shows a Visual Studio Code interface with the following details:

- File Explorer:** Shows a project structure under "NODEJSTOACHUGOKU" with folders "BasicES6", "BasicNodeJS", "1\_what\_is\_node\_js", "2\_modules", and "3\_builtin\_modules". The file "index.js" is selected.
- Code Editor:** Displays the following code in "index.js":

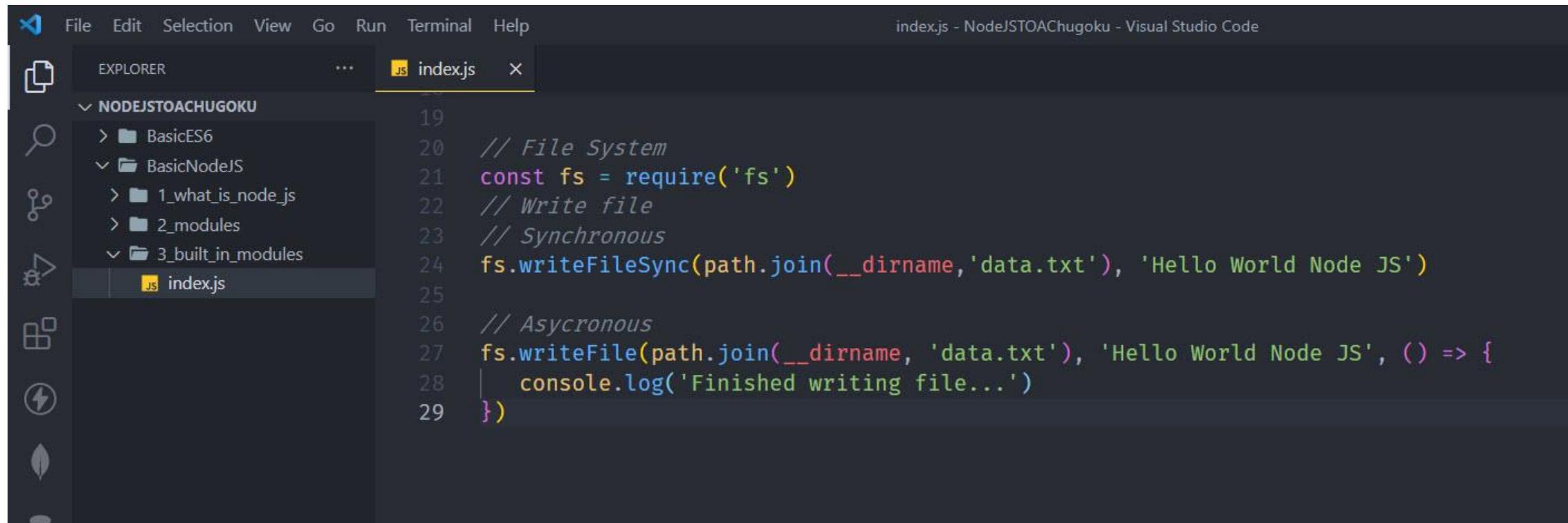
```
// Node.js Built-in Modules
// -----
// 1. File System (fs)
// 2. Path (path)
// 3. Operating System (os)
// 4. Events (events)
// 5. HTTP (http)
// -----
// Path Module
const path = require('path')

console.log(path.basename(__filename))
console.log(path.dirname(__filename));
console.log(path.extname(__filename));
console.log(path.parse(__filename));
console.log(path.join(__dirname, 'utils.js'));
```

- Terminal:** Shows the output of running the script:

```
H:\NodeJSTOACHugoku\BasicNodeJS\3_builtin_modules>node index.js
index.js
H:\NodeJSTOACHugoku\BasicNodeJS\3_builtin_modules
.js
{
  root: 'H:\\\\',
  dir: 'H:\\NodeJSTOACHugoku\\\\BasicNodeJS\\\\3_builtin_modules',
  base: 'index.js',
  ext: '.js',
  name: 'index'
}
H:\NodeJSTOACHugoku\BasicNodeJS\3_builtin_modules\utils.js
```

# Exercise 3: Built-in Module (File)

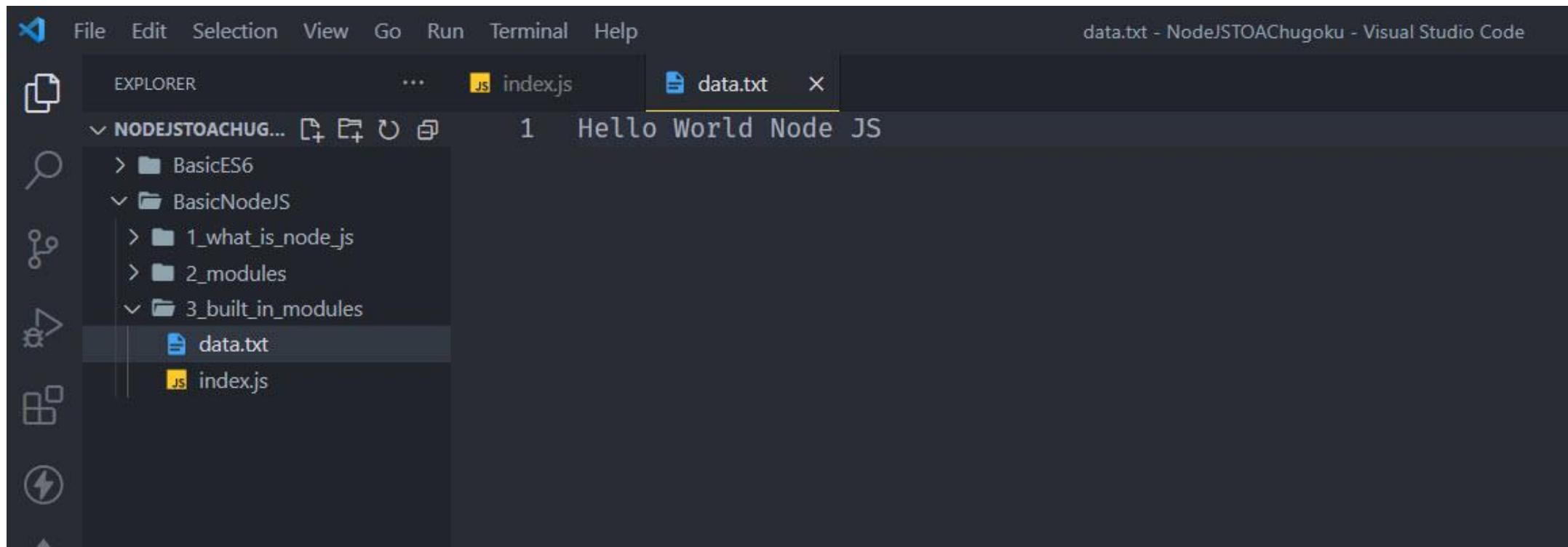


The screenshot shows a Visual Studio Code interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help
- Title Bar:** index.js - NodeJSTOACHUGOKU - Visual Studio Code
- Explorer:** Shows a project structure under 'NODEJSTOACHUGOKU':
  - BasicES6
  - BasicNodeJS
    - 1\_what\_is\_node.js
    - 2\_modules
    - 3\_builtin\_modules
      - index.js
- Code Editor:** The 'index.js' file is open, displaying the following code:

```
19
20 // File System
21 const fs = require('fs')
22 // Write file
23 // Synchronous
24 fs.writeFileSync(path.join(__dirname, 'data.txt'), 'Hello World Node JS')
25
26 // Asynchronous
27 fs.writeFile(path.join(__dirname, 'data.txt'), 'Hello World Node JS', () => {
28   console.log('Finished writing file...')
29 })
```

# Exercise 3: Built-in Module (File)



# Exercise 3: Built-in Module (File)

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL
H:\NodeJSTOACHugoku\BasicNodeJS\3_builtin_modules>node index.js
index.js
H:\NodeJSTOACHugoku\BasicNodeJS\3_builtin_modules
.js
{
  root: 'H:\\',
  dir: 'H:\\NodeJSTOACHugoku\\BasicNodeJS\\3_builtin_modules',
  base: 'index.js',
  ext: '.js',
  name: 'index'
}
H:\NodeJSTOACHugoku\BasicNodeJS\3_builtin_modules\utils.js
Finished writing file...

H:\NodeJSTOACHugoku\BasicNodeJS\3_builtin_modules>
```

# Exercise 3: Built-in Module (Read File)

The screenshot shows a Visual Studio Code interface. The left sidebar has icons for File, Edit, Selection, View, Go, Run, Terminal, and Help. The Explorer sidebar shows a project structure under 'NODEJSTOACHUGOKU': 'BasicES6', 'BasicNodeJS' (which contains '1\_what\_is\_node.js' and '2\_modules'), and '3\_builtin\_modules' (which contains 'data.txt' and 'index.js'). The 'index.js' file is selected in the Explorer. The main editor area shows the following code:

```
19
20 // File System
21 const fs = require('fs')
22 // Write file
23 // Synchronous
24 // fs.writeFileSync(path.join(__dirname, 'data.txt'), 'Hello World Node JS')
25
26 // Asynchronous
27 // fs.writeFile(path.join(__dirname, 'data.txt'), 'Hello World Node JS', () => {
28 //   console.log('Finished writing file...')
29 // })
30
31 // Read file
32 console.log(fs.readFileSync(path.join(__dirname, 'data.txt'), 'utf8'))
33
```

The bottom navigation bar includes 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', and 'TERMINAL'. The terminal window at the bottom shows the command 'node index.js' being run and the output 'Hello World Node JS'.

# Exercise 3: Built-in Module (Operating System)

The screenshot shows a Visual Studio Code interface with the following details:

- File Explorer:** Shows a project structure under "NODEJSTOACHUGOKU" with folders "BasicES6", "BasicNodeJS", "1\_what\_is\_node.js", "2\_modules", and "3\_builtin\_modules". Inside "3\_builtin\_modules", there are files "data.txt" and "index.js".
- Code Editor:** The file "index.js" is open, displaying the following code:

```
// Operating System (os)
const os = require('os')
// show cpu
console.log(os.cpus())
console.log(os.homedir());
console.log(os.uptime());
console.log(os.platform());
console.log(os.release());
console.log(os.arch());
console.log(os.type());
console.log(os.loadavg());
console.log(os.networkInterfaces());
```
- Terminal:** The terminal window shows the command "node index.js" being run, followed by the output:

```
H:\NodeJSTOACHUGOKU\BasicNodeJS\3_builtin_modules>node index.js
[ {
  {
    model: 'Intel(R) Core(TM) i7-8700 CPU @ 3.20GHz',
    speed: 3192,
    times: {
      user: 28976375,
      nice: 0,
      sys: 36997046,
      idle: 500769000,
      irq: 11190953
    },
    {
      model: 'Intel(R) Core(TM) i7-8700 CPU @ 3.20GHz',
      speed: 3192,
      times: {
        user: 28976375,
        nice: 0,
        sys: 36997046,
        idle: 500769000,
        irq: 11190953
      }
    }
  }
]
```

# Exercise 3: Built-in Module (Events)

The screenshot shows a Visual Studio Code interface with the following details:

- File Explorer:** Shows a project structure under "NODEJSTOACHUGOKU". The "3\_built\_in\_modules" folder contains "index.js" and "data.txt".
- Editor:** The "index.js" file is open, displaying code that uses the built-in `events` module to handle an "online" event.
- Terminal:** At the bottom, the terminal window shows the command `node index.js` being run, followed by five instances of the output `A new user has connected`.

```
// Events (events)
const events = require('events')
const EventEmitter = events.EventEmitter;
const connect = new EventEmitter();

connect.on('online', () => {
  console.log('A new user has connected');
})

connect.emit('online')
connect.emit('online')
connect.emit('online')
connect.emit('online')
connect.emit('online')
```

```
H:\NodeJSTOACHUGOKU\BasicNodeJS\3_built_in_modules>node index.js
A new user has connected
```

What is Node.js ?

Modules

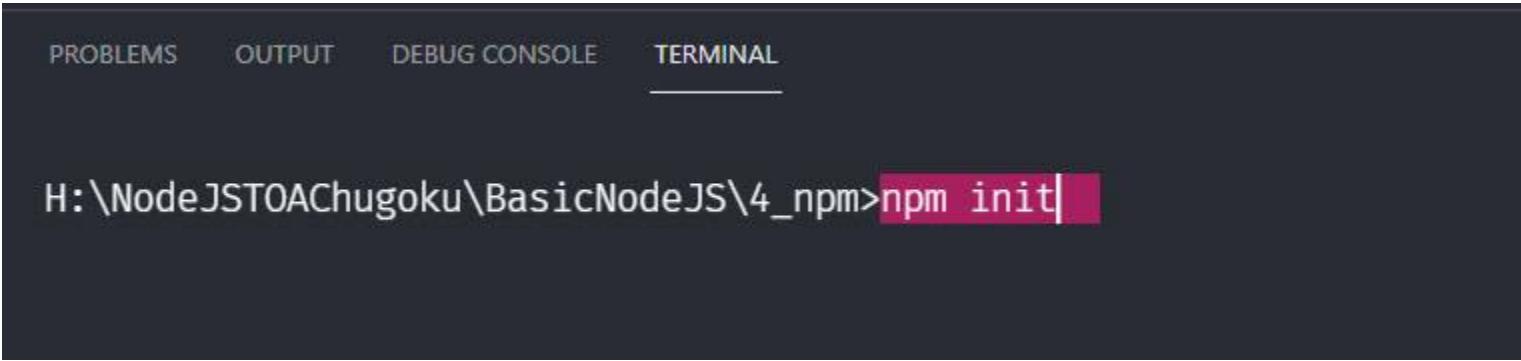
Built-in Modules

Package Manager

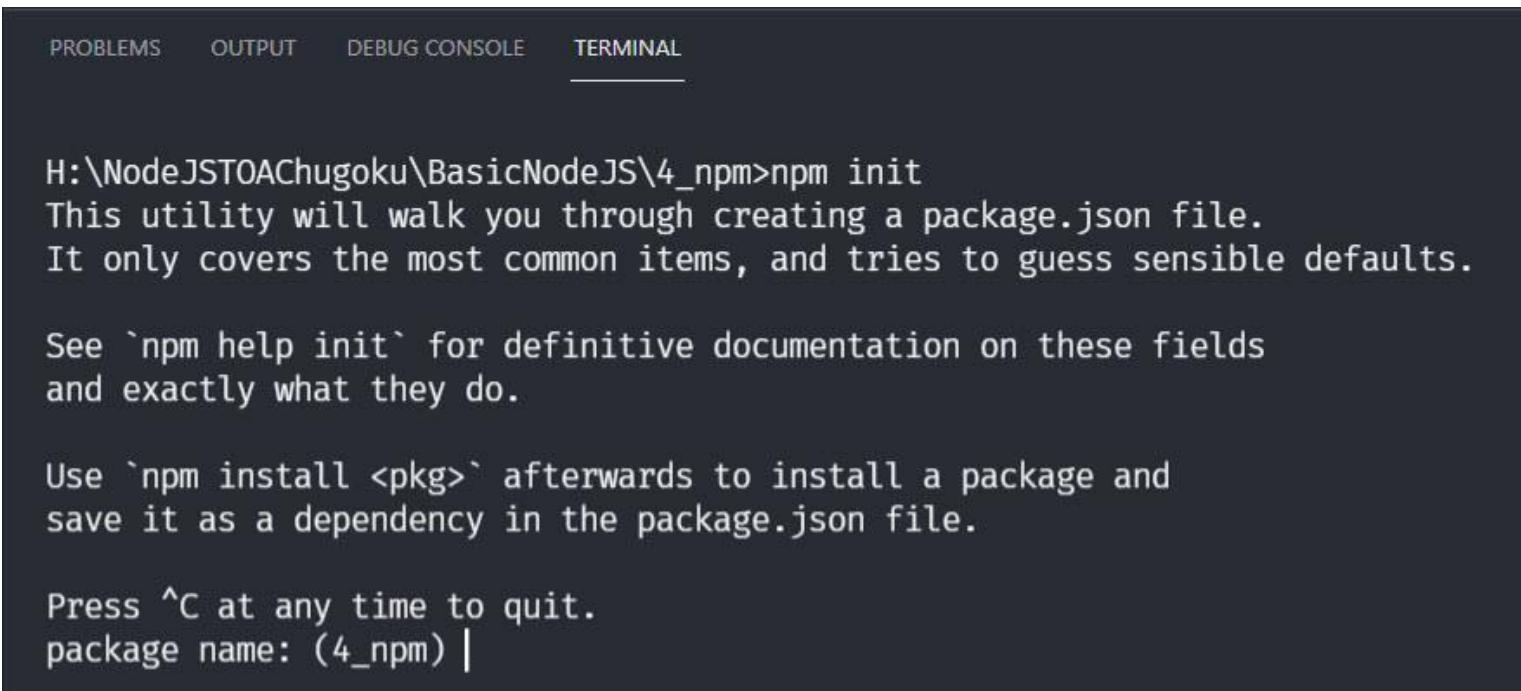
Web Server



# Create package.json



H:\NodeJSTOAChugoku\BasicNodeJS\4\_npm>npm init|



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

H:\NodeJSTOAChugoku\BasicNodeJS\4_npm>npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help init` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (4_npm) |
```

# Create package.json

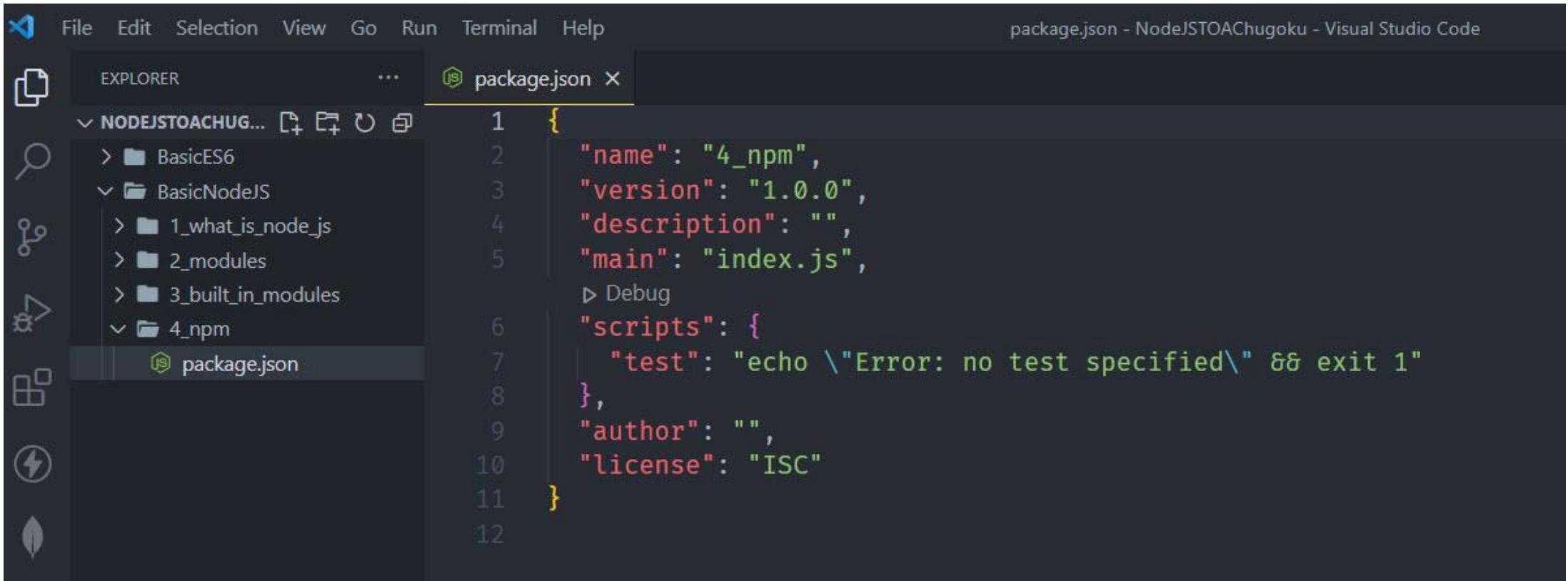
The screenshot shows a terminal window with a dark background and light-colored text. At the top, there are tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, and TERMINAL, with TERMINAL being the active tab. The terminal output is as follows:

```
package name: (4_npm)
version: (1.0.0)
description:
entry point: (index.js)
test command:
git repository:
keywords:
author:
license: (ISC)
About to write to H:\NodeJSTOAChugoku\BasicNodeJS\4_npm\package.json:

{
  "name": "4_npm",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC"
}

Is this OK? (yes) yes|
```

# Create package.json



The screenshot shows the Visual Studio Code interface. The title bar reads "package.json - NodeJSTOACHugoku - Visual Studio Code". The left sidebar (Explorer) shows a project structure under "NODEJSTOACHUGOKU": "BasicES6", "BasicNodeJS" (which contains "1\_what\_is\_node\_js", "2\_modules", "3\_builtin\_modules", and "4\_npm"), and a selected "package.json" file. The main editor area displays the following JSON code:

```
1 {  
2   "name": "4_npm",  
3   "version": "1.0.0",  
4   "description": "",  
5   "main": "index.js",  
6   "scripts": {  
7     "test": "echo \\\"Error: no test specified\\\" && exit 1"  
8   },  
9   "author": "",  
10  "license": "ISC"  
11}  
12
```

# Install package

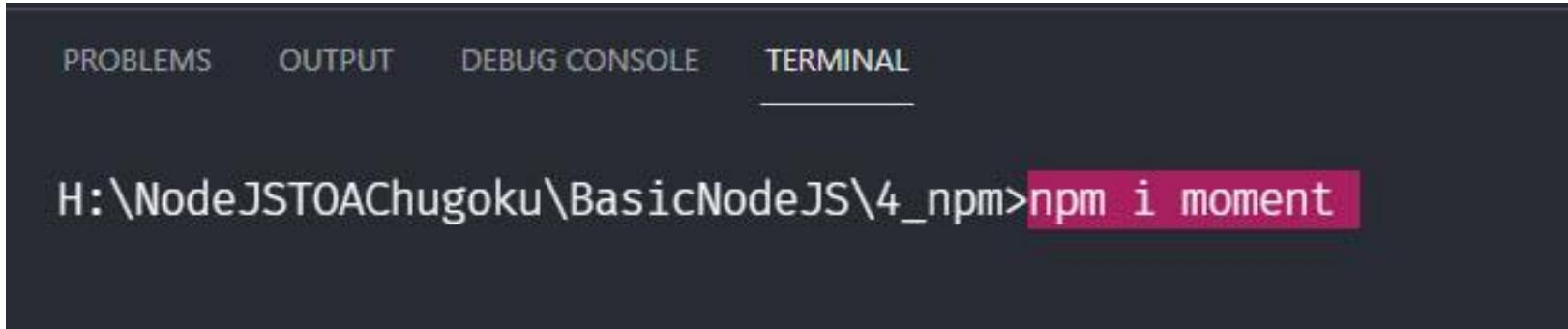
The terminal window shows the command:

```
H:\NodeJSTOACHugoku\BasicNodeJS\4_npm>npm install --save lodash
```

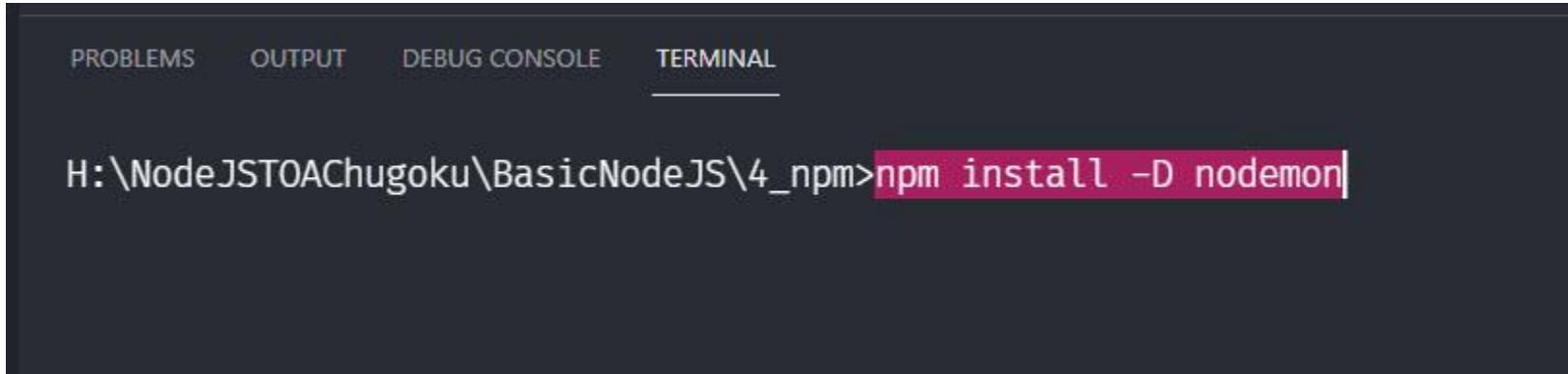
The code editor displays the package.json file with the 'lodash' dependency added:

```
1 {  
2   "name": "4_npm",  
3   "version": "1.0.0",  
4   "description": "",  
5   "main": "index.js",  
6   "scripts": {  
7     "test": "echo \\\"Error: no test specified\\\" && exit 1"  
8   },  
9   "author": "",  
10  "license": "ISC",  
11  "dependencies": {  
12    "lodash": "^4.17.21"  
13  }  
14}  
15}
```

# Install package

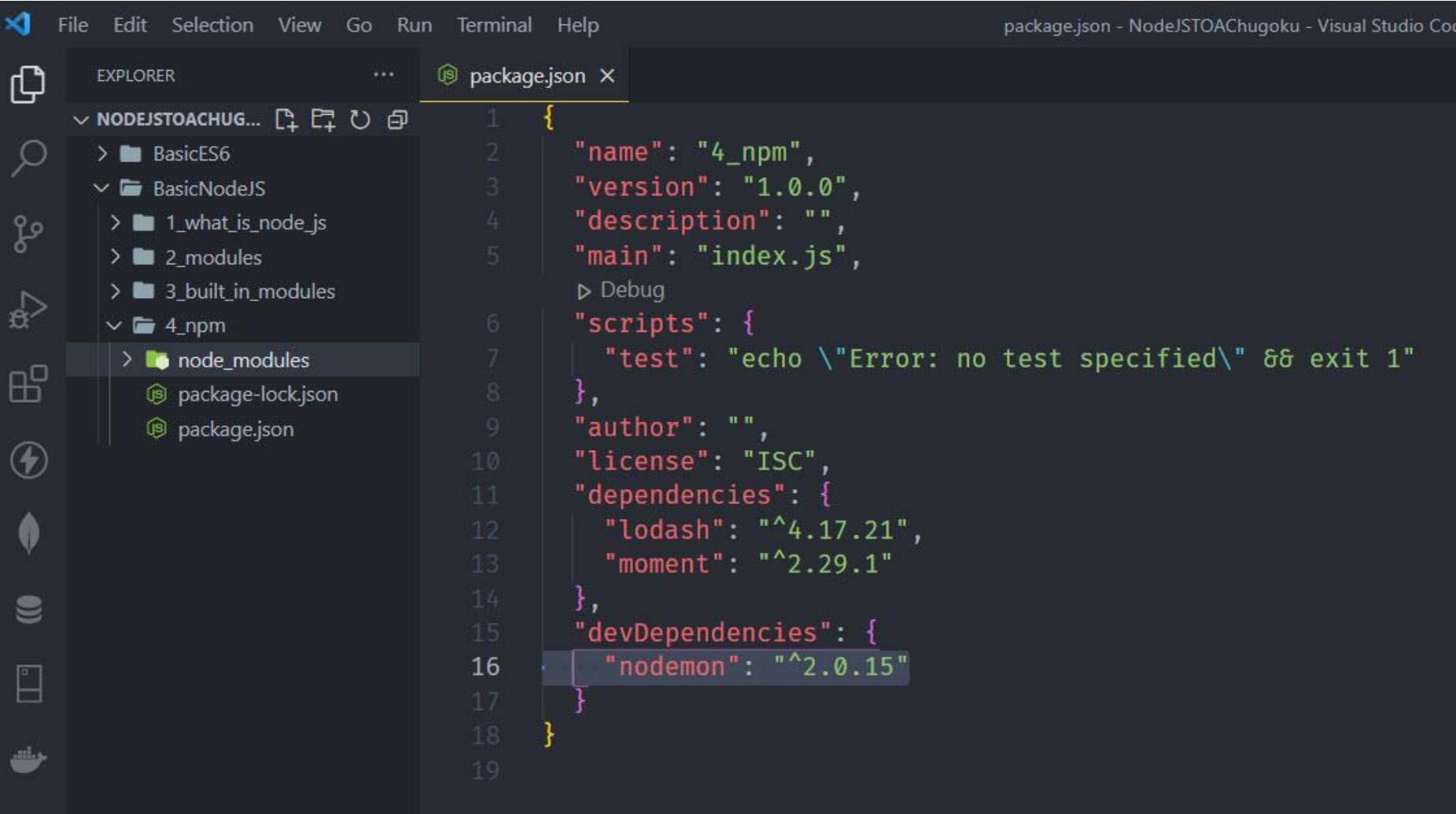


A screenshot of a terminal window from a code editor. The tabs at the top are PROBLEMS, OUTPUT, DEBUG CONSOLE, and TERMINAL, with TERMINAL underlined. In the terminal area, the command `H:\NodeJSTOACHugoku\BasicNodeJS\4_npm>npm i moment` is shown, with the entire command highlighted in a pink rectangle.



A screenshot of a terminal window from a code editor. The tabs at the top are PROBLEMS, OUTPUT, DEBUG CONSOLE, and TERMINAL, with TERMINAL underlined. In the terminal area, the command `H:\NodeJSTOACHugoku\BasicNodeJS\4_npm>npm install -D nodemon` is shown, with the entire command highlighted in a pink rectangle.

# Install package



The screenshot shows the Visual Studio Code interface with the title bar "package.json - NodeJSTOAChugoku - Visual Studio Code". The left sidebar is the Explorer view, showing a project structure under "NODEJSTOAChugoku": "BasicES6", "BasicNodeJS", "1\_what\_is\_node\_js", "2\_modules", "3\_built\_in\_modules", and "4\_npm". Inside "4\_npm", there are three files: "node\_modules", "package-lock.json", and "package.json". The "package.json" file is selected and open in the main editor area. The code in the editor is:

```
1  {
2      "name": "4_npm",
3      "version": "1.0.0",
4      "description": "",
5      "main": "index.js",
6      "scripts": {
7          "test": "echo \\\"Error: no test specified\\\" && exit 1"
8      },
9      "author": "",
10     "license": "ISC",
11     "dependencies": {
12         "lodash": "^4.17.21",
13         "moment": "^2.29.1"
14     },
15     "devDependencies": {
16         "nodemon": "^2.0.15"
17     }
18 }
```

The line "nodemon": "^2.0.15" is highlighted with a blue selection bar.

# Install nodemon in globally

A screenshot of the VS Code interface showing the Terminal tab selected. The terminal window displays the command `npm install -g nodemon` being typed into it.

A screenshot of the VS Code interface showing the Terminal tab selected. The terminal window displays the command `nodemon -v` followed by the output `2.0.15`.

# Use Nodemon to run file

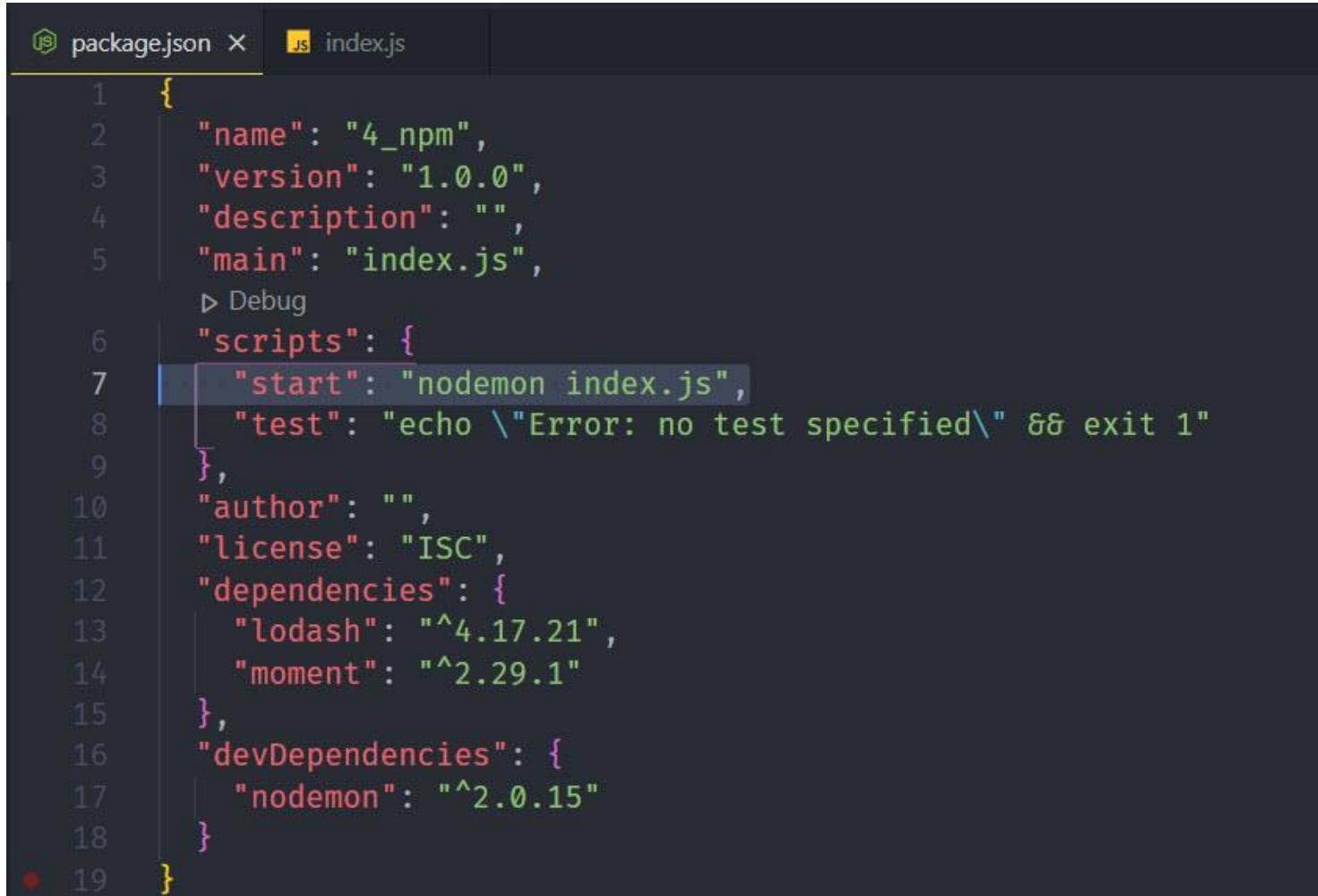
The screenshot shows a VS Code interface with the following details:

- EXPLORER** sidebar: Shows a project structure under "NODEJSTOACHUGOKU" with folders: BasicES6, BasicNodeJS (containing 1\_what\_is\_node\_js, 2\_modules, 3\_builtin\_modules), 4\_npm (containing node\_modules and index.js), package-lock.json, and package.json.
- PACKAGE.JSON**: The active tab in the top bar.
- index.js**: The active tab in the code editor, containing the following code:

```
1 // Node Package Manager
2 // https://www.npmjs.com/
3
4 console.log('Hello Nodemon');
5
```
- TERMINAL**: The active tab in the bottom bar.
- Terminal Output**:

```
H:\NodeJSTOACHugoku\BasicNodeJS\4_npm>nodemon index.js
[nodemon] 2.0.15
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): ***!
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node index.js`
Hello Nodemon
[nodemon] clean exit - waiting for changes before restart
```

# Use Nodemon to run file



The screenshot shows a code editor with two tabs: "package.json" and "index.js". The "package.json" tab is active, displaying the following JSON configuration:

```
1  {
2      "name": "4_npm",
3      "version": "1.0.0",
4      "description": "",
5      "main": "index.js",
6      "scripts": {
7          "start": "nodemon index.js",
8          "test": "echo \"Error: no test specified\" && exit 1"
9      },
10     "author": "",
11     "license": "ISC",
12     "dependencies": {
13         "lodash": "^4.17.21",
14         "moment": "^2.29.1"
15     },
16     "devDependencies": {
17         "nodemon": "^2.0.15"
18     }
19 }
```

# Use Nodemon to run file

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL
H:\NodeJSTOChugoku\BasicNodeJS\4_npm>npm start

> 4_npm@1.0.0 start H:\NodeJSTOChugoku\BasicNodeJS\4_npm
> nodemon index.js

[nodemon] 2.0.15
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): ***!
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node index.js`
Hello Nodemon
สวัสดี Node.JS Programming
[nodemon] clean exit - waiting for changes before restart
```

# What is Node.js ?

Modules

Built-in Modules

Package Manager

Web Server



# Exercise 5: Web Server with http module

The screenshot shows a Visual Studio Code interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Title Bar:** demohttp.js - NodeJSTOACHugoku - Visual Studio Code.
- Explorer:** Shows a tree view of a project named "NODEJSTOACHUGOKU". The "BasicNodeJS" folder contains subfolders "1\_what\_is\_node.js", "2\_modules", and "3\_builtin\_modules". Inside "3\_builtin\_modules", there are files "data.txt", "demohttp.js" (which is selected), and "index.js".
- Code Editor:** Displays the content of the "demohttp.js" file, which is a Node.js script using the http module to create a web server. The code defines a server object, handles requests for '/' and '/contact', logs client connections, and listens on port 3000.

```
const http = require('http');
const server = http.createServer(function(req, res){
  var obj = {
    name: "Samit",
    email: "samit@email.com",
    tel: "089-77389202"
  };
  if (req.url === '/') {
    res.write('You are at home page');
    res.end();
  }
  if (req.url === '/contact') {
    res.write(JSON.stringify(obj));
    res.end();
  }
});
server.on('connection', function(socket){
  console.log('Client connected');
});
server.listen(3000);
console.log('Listening from port 3000');
```

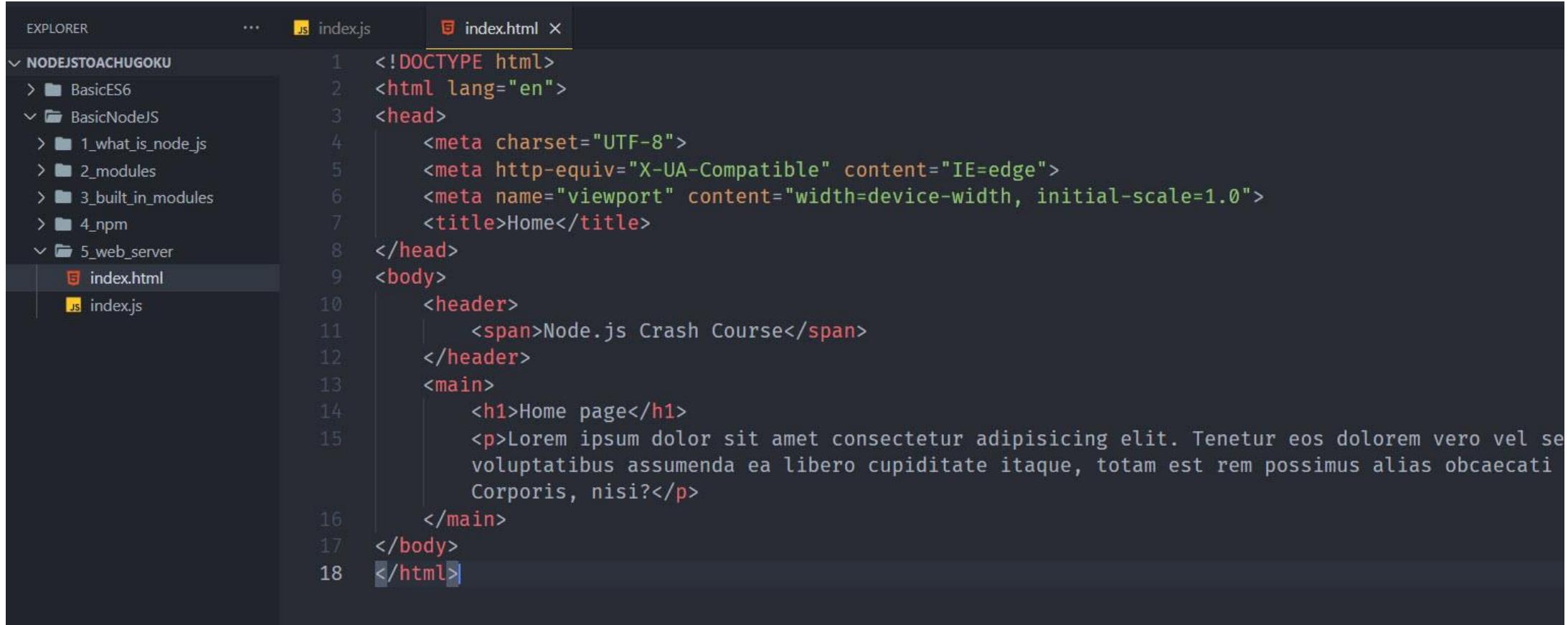
# Exercise 5: Web Server with http module

The screenshot shows a Visual Studio Code interface with the following details:

- File Menu:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Title Bar:** index.js - NodeJSTOACHugoku - Visual Studio C
- Explorer:** Shows a tree view of a project named "NODEJSTOACHUGOKU". The "BasicNodeJS" folder contains sub-folders: "1\_what\_is\_node\_js", "2\_modules", "3\_builtin\_modules", "4\_npm", and "5\_web\_server". The "index.js" file under "5\_web\_server" is selected.
- Code Editor:** Displays the following code in "index.js":

```
const http = require('http')
http.createServer((req, res) => {
  res.setHeader('Content-Type', 'text/html')
  res.writeHead(200)
  res.write('<h1>Hello</h1>')
  res.end()
}).listen(3000)
```
- Terminal:** Shows the command: H:\NodeJSTOACHugoku\BasicNodeJS\5\_web\_server>node index.js
- Browser Preview:** A small browser window shows the output: Hello

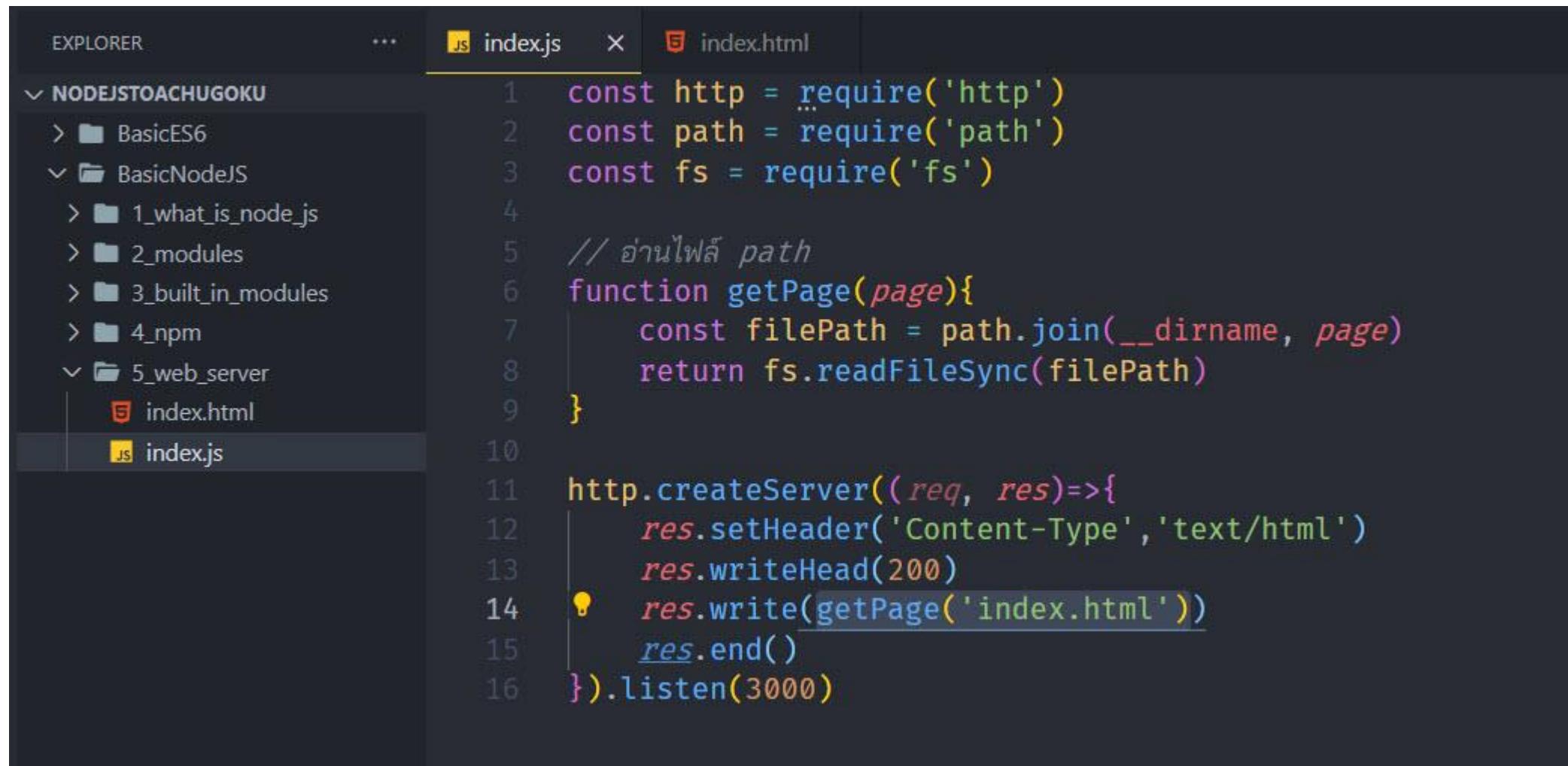
# Exercise 5: Web Server with http module



The screenshot shows a dark-themed code editor in VS Code. The left sidebar displays a project structure under 'NODESTOACHUGOKU' with sub-folders 'BasicES6', 'BasicNodeJS' (containing '1\_what\_is\_node.js', '2\_modules', '3\_built\_in\_modules', '4\_npm'), '5\_web\_server' (containing 'index.html' and 'index.js'), and a 'RECENTS' section. The right pane shows the content of 'index.html'. The code is a basic HTML document with a title, header, body, and main content.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Home</title>
</head>
<body>
    <header>
        <span>Node.js Crash Course</span>
    </header>
    <main>
        <h1>Home page</h1>
        <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Tenetur eos dolorem vero vel se voluptatibus assumenda ea libero cupiditate itaque, totam est rem possimus alias obcaecati Corporis, nisi?</p>
    </main>
</body>
</html>
```

# Exercise 5: Web Server with http module



The screenshot shows a code editor in VS Code with the following file structure:

- EXPLORER: NODESTOACHUGOKU
  - BasicES6
  - BasicNodeJS
    - 1\_what\_is\_node\_js
    - 2\_modules
    - 3\_builtin\_modules
    - 4\_npm
  - 5\_web\_server
    - index.html
    - index.js

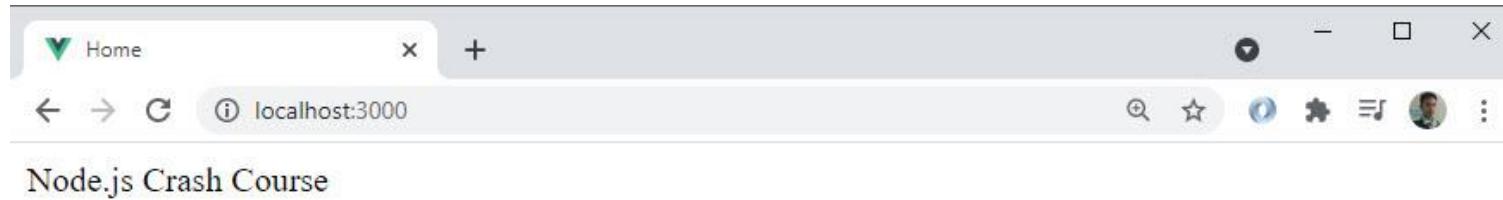
The active file is `index.js`, which contains the following code:

```
const http = require('http')
const path = require('path')
const fs = require('fs')

// อ่านไฟล์ path
function getPage(page){
    const filePath = path.join(__dirname, page)
    return fs.readFileSync(filePath)
}

http.createServer((req, res)=>{
    res.setHeader('Content-Type', 'text/html')
    res.writeHead(200)
    res.write(getPage('index.html'))
    res.end()
}).listen(3000)
```

# Exercise 5: Web Server with http module



## Home page

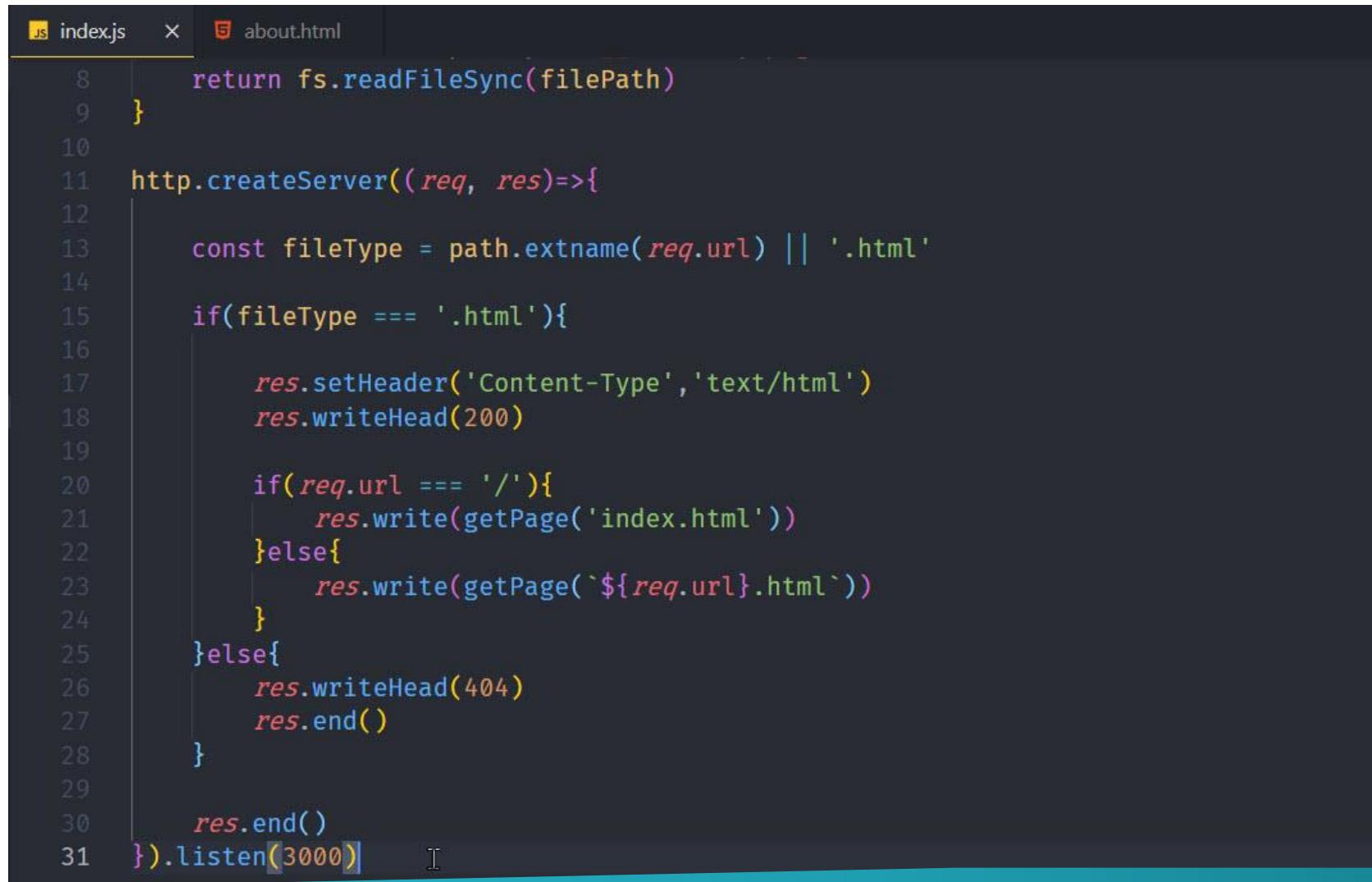
Lorem ipsum dolor sit amet consectetur adipisicing elit. Tenetur eos dolorem vero vel sequi voluptatibus assumenda ea libero cupiditate itaque, totam est rem possimus alias obcaecati omnis veniam! Corporis, nisi?

# Exercise 5: Web Server with http module

The screenshot shows a dark-themed interface of the Visual Studio Code code editor. On the left, the Explorer sidebar displays a project structure under the folder 'NODEJS TO A CHUGOKU'. The '5\_web\_server' folder contains three files: 'about.html', 'index.html', and 'index.js'. The 'about.html' file is currently open in the main editor area. The code is an HTML document with the following content:

```
1 <!DOCTYPE html>
2 <html lang="en">
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     <title>About</title>
8   </head>
9   <body>
10    <header>
11      <span>Node.js Crash Course</span>
12    </header>
13    <main>
14      <h1>About page</h1>
15      <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Tenetur eos dolorem  
voluptatibus assumenda ea libero cupiditate itaque, totam est rem possimus alias  
Corporis, nisi?</p>
16    </main>
17  </body>
18 </body>
19 </html>
```

# Exercise 5: Web Server with http module

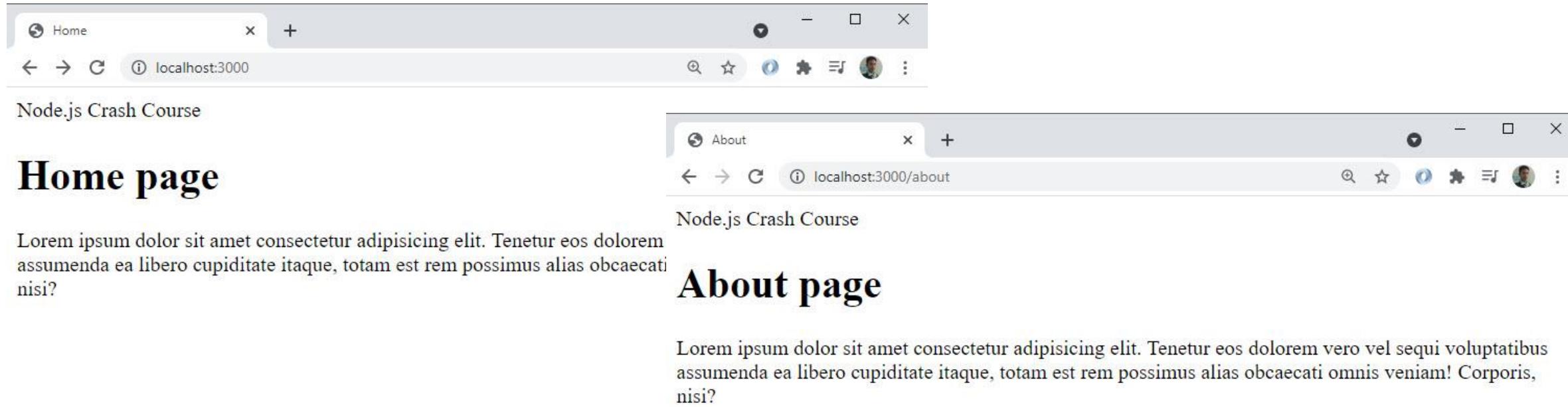


The image shows a code editor interface with two tabs: 'index.js' and 'about.html'. The 'index.js' tab is active, displaying the following code:

```
index.js  ✘ about.html

8     return fs.readFileSync(filePath)
9 }
10
11 http.createServer((req, res)=>{
12
13     const fileType = path.extname(req.url) || '.html'
14
15     if(fileType === '.html'){
16
17         res.setHeader('Content-Type','text/html')
18         res.writeHead(200)
19
20         if(req.url === '/'){
21             res.write(getPage('index.html'))
22         }else{
23             res.write(getPage(` ${req.url}.html`))
24         }
25     }else{
26         res.writeHead(404)
27         res.end()
28     }
29
30     res.end()
31 }).listen(3000)
```

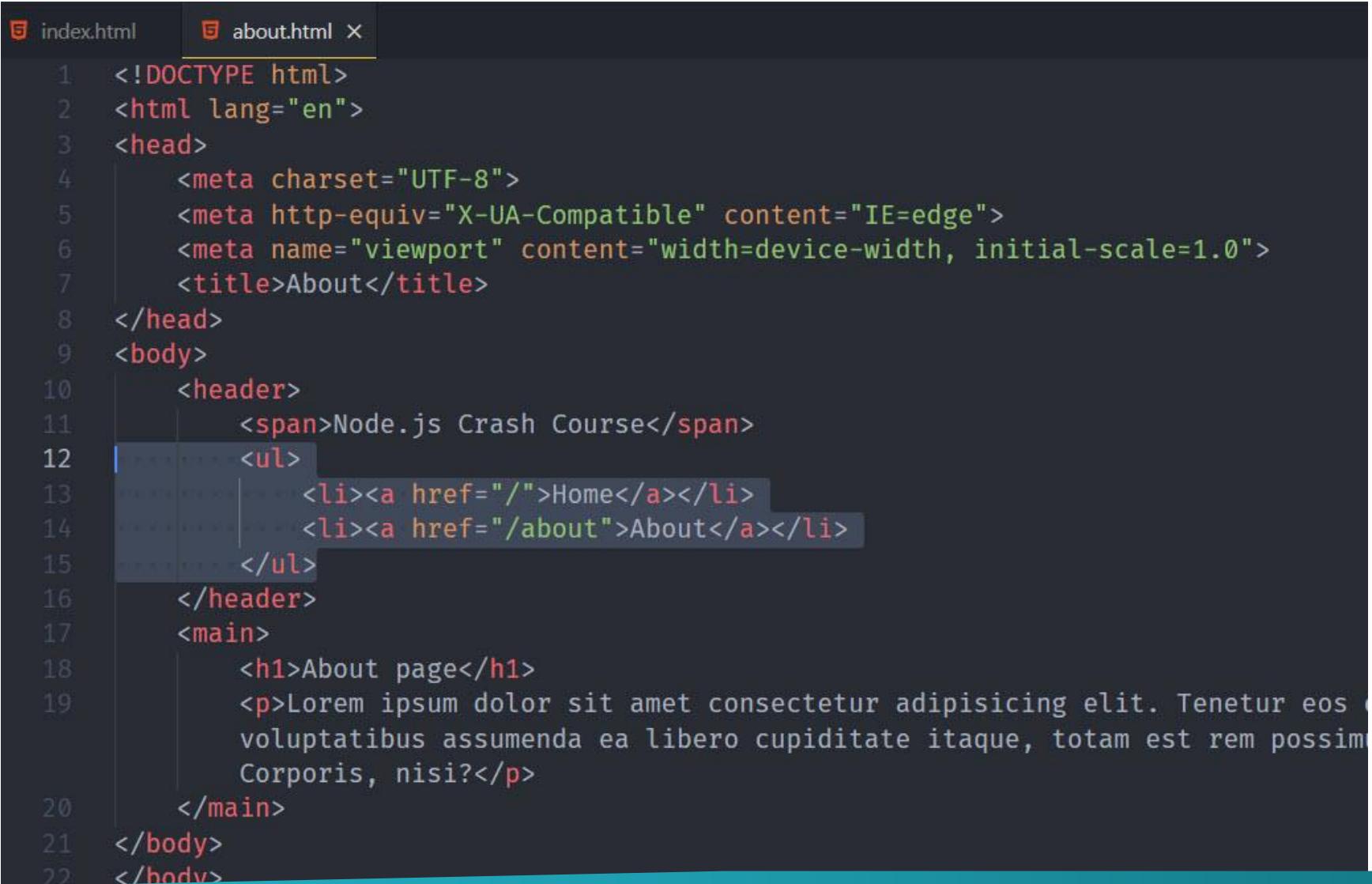
# Exercise 5: Web Server with http module



# Exercise 5: Web Server with http module

```
index.html × about.html
 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   <title>Home</title>
 8 </head>
 9 <body>
10   <header>
11     <span>Node.js Crash Course</span>
12     <ul>
13       <li><a href="/">Home</a></li>
14       <li><a href="/about">About</a></li>
15     </ul>
16   </header>
17   <main>
18     <h1>Home page</h1>
19     <p>Lorem ipsum dolor sit amet consectetur adipisicing elit.  
voluptatibus assumenda ea libero cupiditate itaque, totam es  
Corporis, nisi?</p>
20   </main>
21 </body>
22 </html>
```

# Exercise 5: Web Server with http module



```
index.html      about.html ×
1  <!DOCTYPE html>
2  <html lang="en">
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      <title>About</title>
8  </head>
9  <body>
10     <header>
11         <span>Node.js Crash Course</span>
12         <ul>
13             <li><a href="/">Home</a></li>
14             <li><a href="/about">About</a></li>
15         </ul>
16     </header>
17     <main>
18         <h1>About page</h1>
19         <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Tenetur eos
20             voluptatibus assumenda ea libero cupiditate itaque, totam est rem possim
21                 Corporis, nisi?</p>
22     </main>
23 </body>
24 </html>
```

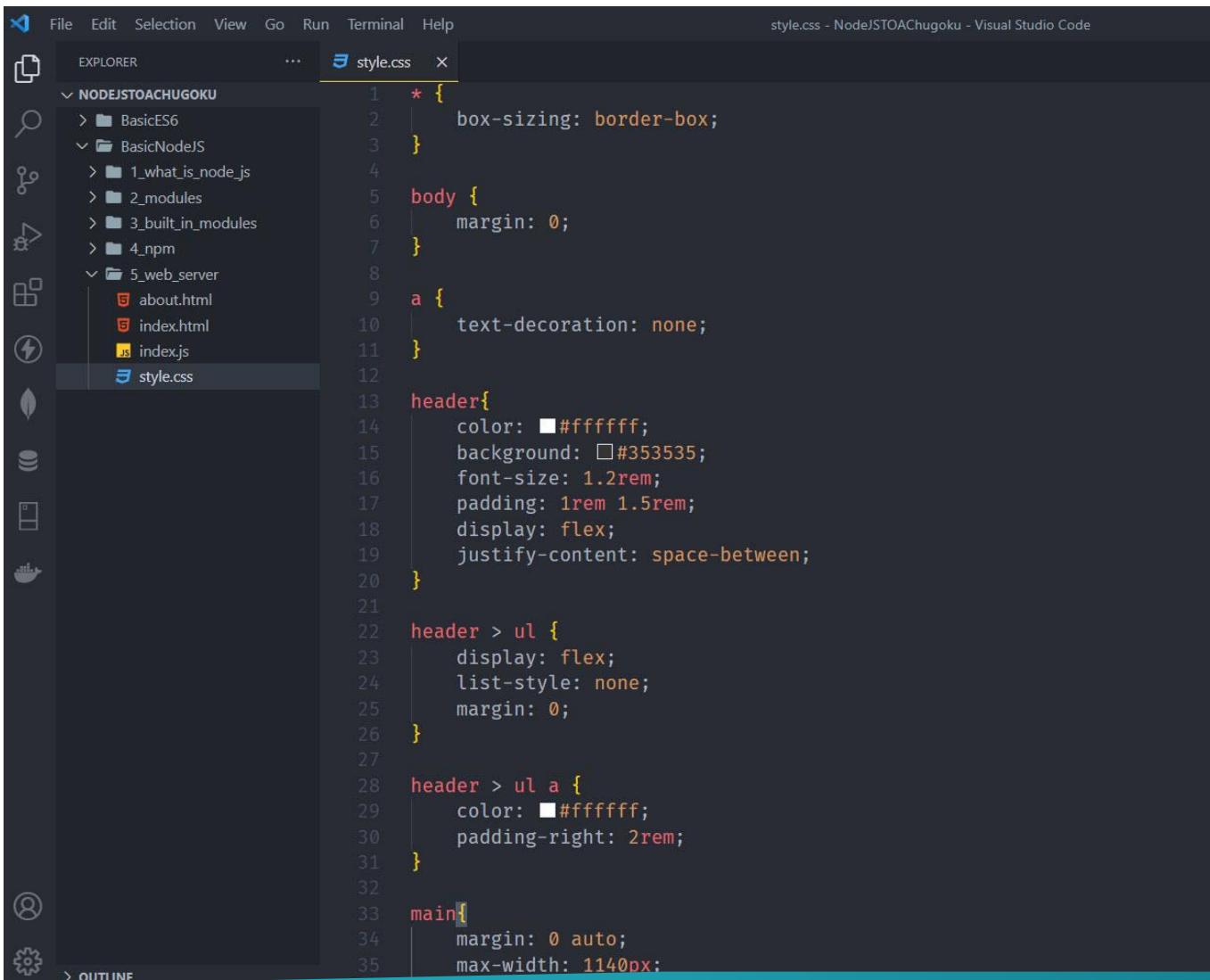
# Exercise 5: Web Server with http module



## About page

Lorem ipsum dolor sit amet consectetur adipisicing elit. Tenetur eos dolorem vero vel sequi voluptatibus assumenda ea libero cupiditate itaque, totam est rem possimus alias obcaecati omnis veniam! Corporis, nisi?

# Exercise 5: Web Server with http module

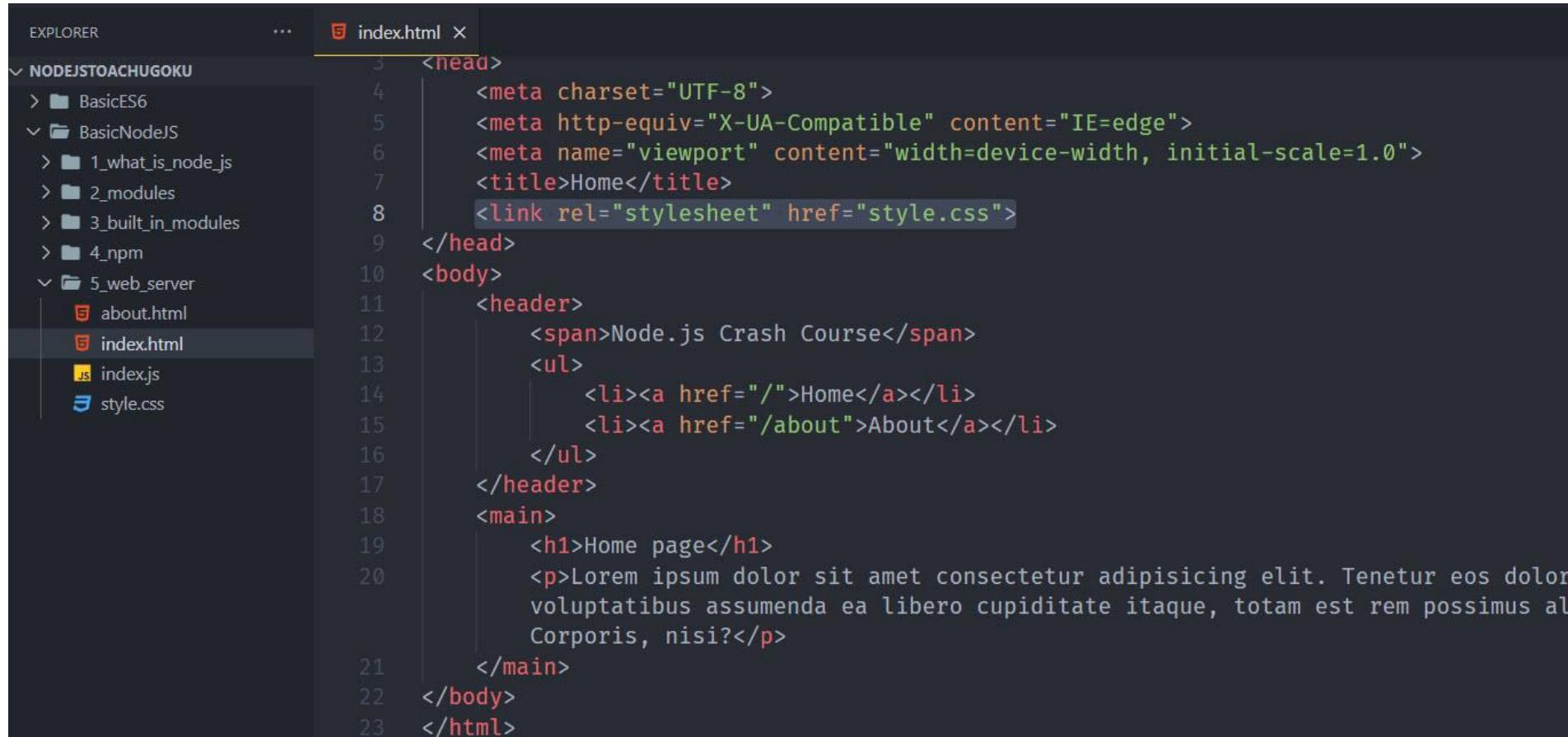


The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows a project structure under the folder "NODEJSTOACHUGOKU". The "style.css" file is selected.
- Terminal:** Shows the command "style.css - NodeJSTOACHugoku - Visual Studio Code".
- Code Editor:** Displays the CSS code for "style.css".

```
* {  
    box-sizing: border-box;  
}  
  
body {  
    margin: 0;  
}  
  
a {  
    text-decoration: none;  
}  
  
header{  
    color: #ffffff;  
    background: #353535;  
    font-size: 1.2rem;  
    padding: 1rem 1.5rem;  
    display: flex;  
    justify-content: space-between;  
}  
  
header > ul {  
    display: flex;  
    list-style: none;  
    margin: 0;  
}  
  
header > ul a {  
    color: #ffffff;  
    padding-right: 2rem;  
}  
  
main{  
    margin: 0 auto;  
    max-width: 1140px;  
}
```

# Exercise 5: Web Server with http module



The screenshot shows the VS Code interface with the following details:

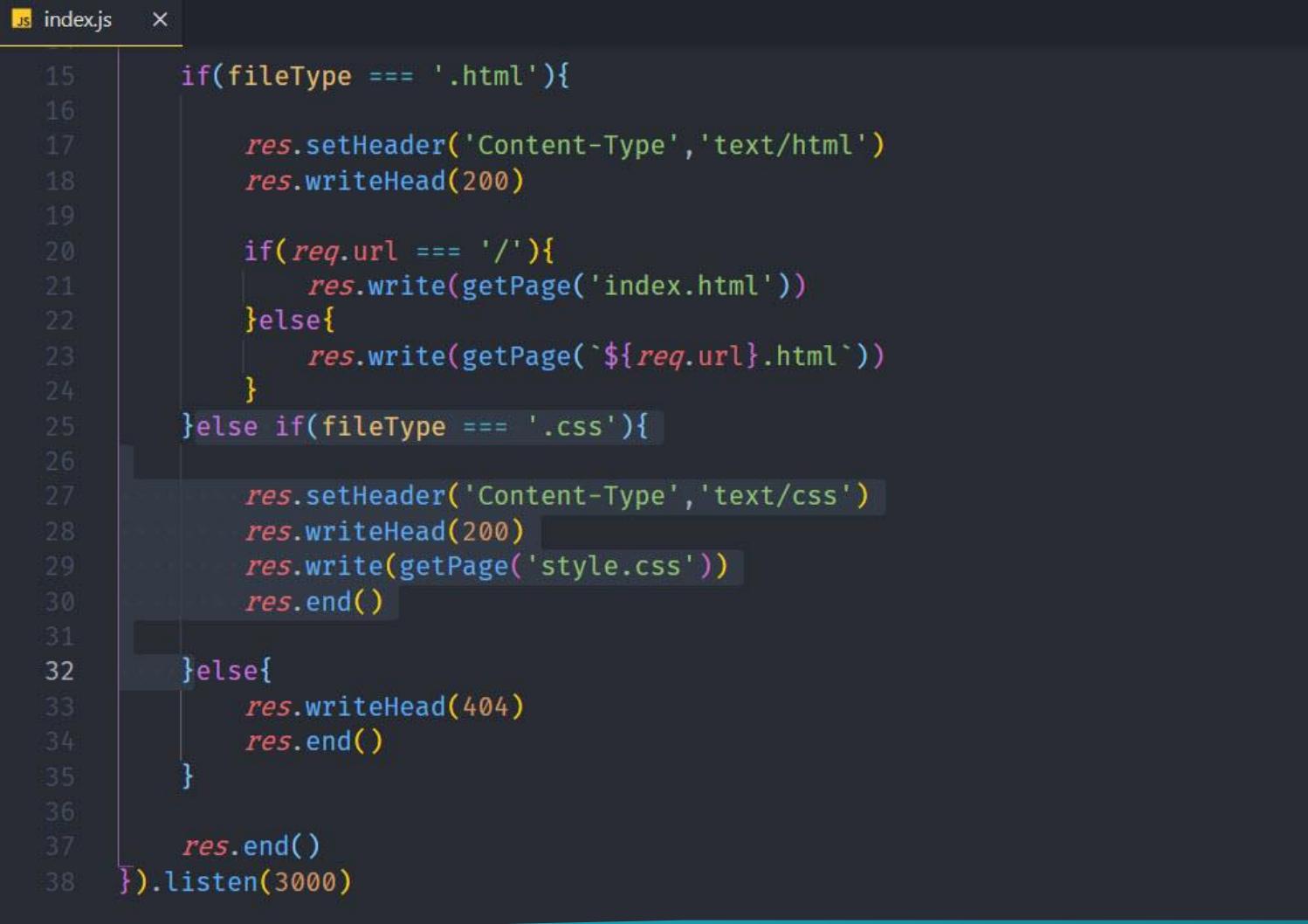
- EXPLORER**: Shows a tree view of the project structure under the folder `NODEJS TO A CHUGOKU`. The `index.html` file is selected in the Explorer.
- index.html**: The active tab in the editor window, showing the following code:

```
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Home</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <header>
    <span>Node.js Crash Course</span>
    <ul>
      <li><a href="/">Home</a></li>
      <li><a href="/about">About</a></li>
    </ul>
  </header>
  <main>
    <h1>Home page</h1>
    <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Tenetur eos dolor  
voluptatibus assumenda ea libero cupiditate itaque, totam est rem possimus al  
Corporis, nisi?</p>
  </main>
</body>
</html>
```

# Exercise 5: Web Server with http module

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>About</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <header>
        <span>Node.js Crash Course</span>
        <ul>
            <li><a href="/">Home</a></li>
            <li><a href="/about">About</a></li>
        </ul>
    </header>
    <main>
        <h1>About page</h1>
        <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Tenetur eos  
voluptatibus assumenda ea libero cupiditate itaque, totam est rem possim  
Corporis, nisi?</p>
    </main>
</body>
</body>
</html>
```

# Exercise 5: Web Server with http module



```
JS index.js x
15     if(fileType === '.html'){
16
17         res.setHeader('Content-Type','text/html')
18         res.writeHead(200)
19
20         if(req.url === '/'){
21             res.write(getPage('index.html'))
22         }else{
23             res.write(getPage(` ${req.url}.html`))
24         }
25     }else if(fileType === '.css'){
26
27         res.setHeader('Content-Type','text/css')
28         res.writeHead(200)
29         res.write(getPage('style.css'))
30         res.end()
31
32     }else{
33         res.writeHead(404)
34         res.end()
35     }
36
37     res.end()
38 }).listen(3000)
```

# Exercise 5: Web Server with http module



## Home page

Lorem ipsum dolor sit amet consectetur adipisicing elit. Tenetur eos dolorem vero vel sequi voluptatibus assumenda ea libero cupiditate itaque, totam est rem possimus alias obcaecati omnis veniam! Corporis, nisi?

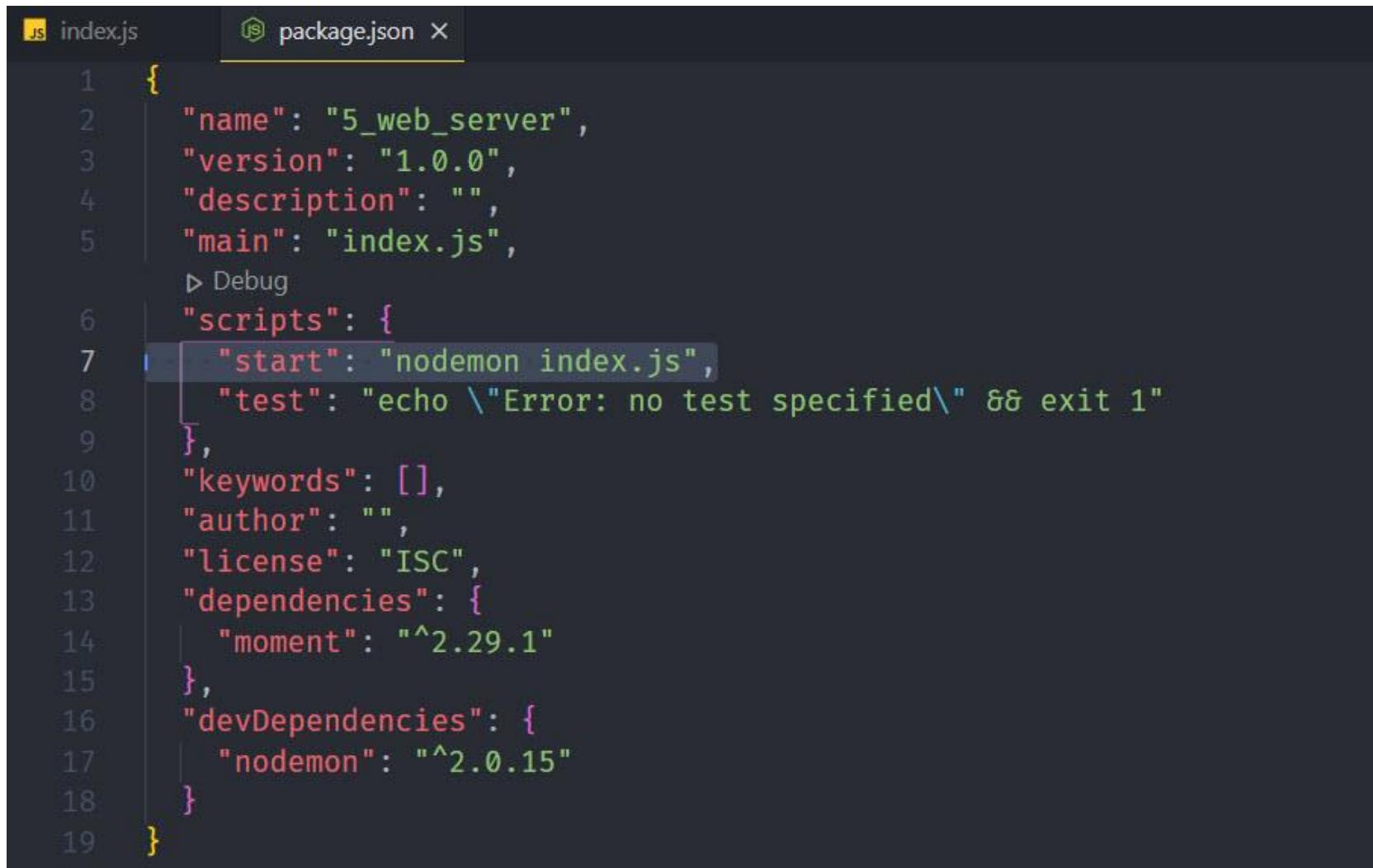
# Exercise 5: Create API

H:\NodeJSTOChugoku\BasicNodeJS\5\_web\_server>npm init -y

H:\NodeJSTOChugoku\BasicNodeJS\5\_web\_server>npm install moment

H:\NodeJSTOChugoku\BasicNodeJS\5\_web\_server>npm install -D nodemon  
[ ..... ] \ fetchMetadata: sill resolveWithNewModule ignore-by-default@1.0.1 checking installable status

# Exercise 5: Create API



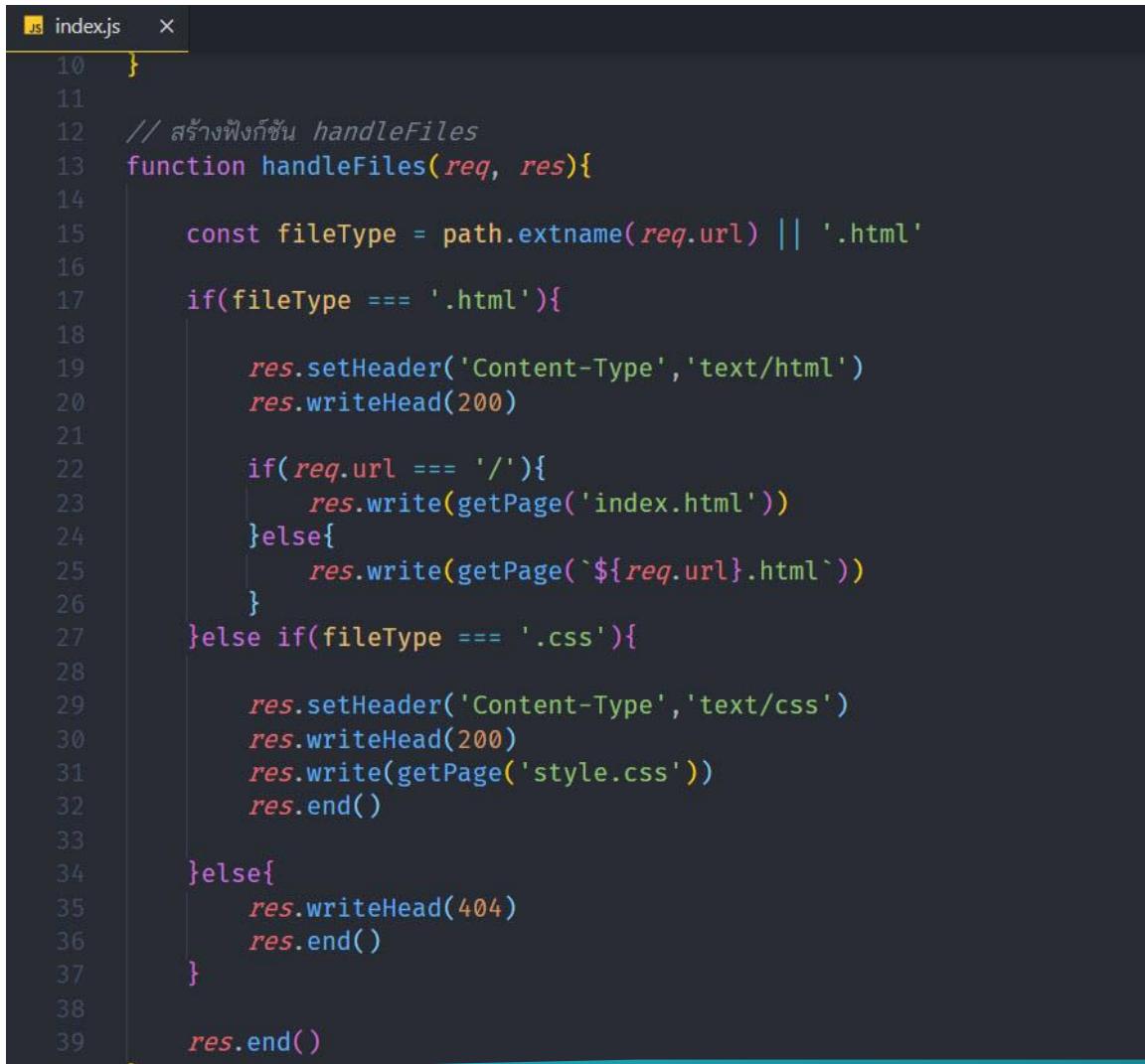
The screenshot shows a code editor with two tabs: index.js and package.json. The package.json tab is active, displaying the following JSON configuration:

```
1  {
2    "name": "5_web_server",
3    "version": "1.0.0",
4    "description": "",
5    "main": "index.js",
6    "scripts": {
7      "start": "nodemon index.js",
8      "test": "echo \"Error: no test specified\" && exit 1"
9    },
10   "keywords": [],
11   "author": "",
12   "license": "ISC",
13   "dependencies": {
14     "moment": "^2.29.1"
15   },
16   "devDependencies": {
17     "nodemon": "^2.0.15"
18   }
19 }
```

## Exercise 5: Create API

```
JS index.js ×
1 const http = require('http')
2 const path = require('path')
3 const fs = require('fs')
4 const moment = require('moment')
5
6 // อ่านไฟล์ path
7 function getPage(page){
8     const filePath = path.join(__dirname, page)
9     return fs.readFileSync(filePath)
10 }
11
```

# Exercise 5: Create API



```
index.js  x
10  }
11
12 // สร้างฟังก์ชัน handleFiles
13 function handleFiles(req, res){
14
15     const fileType = path.extname(req.url) || '.html'
16
17     if(fileType === '.html'){
18
19         res.setHeader('Content-Type', 'text/html')
20         res.writeHead(200)
21
22         if(req.url === '/'){
23             res.write(getPage('index.html'))
24         }else{
25             res.write(getPage(` ${req.url}.html`))
26         }
27     }else if(fileType === '.css'){
28
29         res.setHeader('Content-Type', 'text/css')
30         res.writeHead(200)
31         res.write(getPage('style.css'))
32         res.end()
33
34     }else{
35         res.writeHead(404)
36         res.end()
37     }
38
39     res.end()
40 }
```

## Exercise 5: Create API

```
42 // สร้างฟังก์ชัน getData
43 function getData(url){
44     let data;
45     if(url === '/api/users'){
46         data = [{name:'Samit'}, {name:'Siriwat'}]
47     }else if(url === '/api/posts'){
48         data = [
49             {
50                 title: 'Story 1',
51                 publishedDate: moment().startOf('day').fromNow()
52             },
53             {
54                 title: 'Story 2',
55                 publishedDate: moment().set('month',1).startOf('day').fromNow()
56             }
57         ]
58     }
59     return data
60 }
```

# Exercise 5: Create API

```
62 // สร้างฟังก์ชัน handleAPis
63 function handleAPis(req, res){
64
65     let data = getData(req.url)
66     // console.log(data);
67
68     if(data){
69         res.setHeader('Content-Type', 'application/json')
70         res.write(JSON.stringify(data))
71     }else{
72         res.writeHead(404)
73     }
74
75     res.end()
76 }
77
78
79 http.createServer((req, res)=>{
80     if(req.url.startsWith('/api/')){
81         handleAPis(req, res);
82     }else{
83         handleFiles(req, res);
84     }
85 }).listen(3000)
```

# Exercise 5: Create API

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

```
H:\NodeJSTOACHugoku\BasicNodeJS\5_web_server>npm start  
> 5_web_server@1.0.0 start H:\NodeJSTOACHugoku\BasicNodeJS\5_web_server  
> nodemon index.js  
  
[nodemon] 2.0.15  
[nodemon] to restart at any time, enter `rs`  
[nodemon] watching path(s): *.*  
[nodemon] watching extensions: js,mjs,json  
[nodemon] starting `node index.js`
```

## Exercise 5: Create API



A screenshot of a web browser window displaying a JSON response. The address bar shows 'localhost:3000/api/users'. The page content is a JSON array containing two objects, each with a 'name' field. The 'name' fields are highlighted in green.

```
[  
  - {  
      name: "Samit"  
    },  
  - {  
      name: "Siriwat"  
    }  
]
```

## Exercise 5: Create API



A screenshot of a web browser window titled "localhost:3000/api/posts". The address bar also shows "localhost:3000/api/posts". The page content displays a JSON array of two posts:

```
[  
  - {  
      title: "Story 1",  
      publishedDate: "2 hours ago"  
    },  
  - {  
      title: "Story 2",  
      publishedDate: "9 months ago"  
    }  
]
```

# Exercise 5: Create API

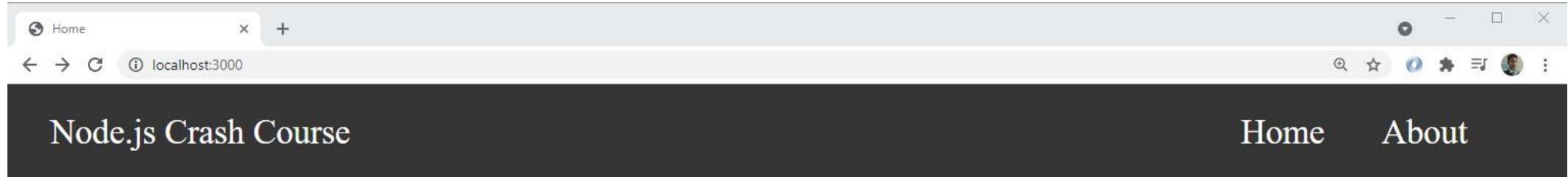
The screenshot shows the VS Code interface with the following details:

- EXPLORER** sidebar: Shows a tree view of the project structure under "NODESTOACHUG...". The "index.html" file is selected.
- index.js**: A JavaScript file containing boilerplate code for a Node.js application.
- index.html**: An HTML file currently open in the editor.

The content of the `index.html` file is as follows:

```
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Home</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <header>
    <span>Node.js Crash Course</span>
    <ul>
      <li><a href="/">Home</a></li>
      <li><a href="/about">About</a></li>
    </ul>
  </header>
  <main>
    <h1>Home page</h1>
    <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Tenetur eos dolorem ver  
voluptatibus assumenda ea libero cupiditate itaque, totam est rem possimus alias ob  
Corporis, nisi?</p>
    <ul>
      <li><a href="/api/users">Users</a></li>
      <li><a href="/api/posts">Posts</a></li>
    </ul>
  </main>
</body>
</html>
```

# Exercise 5: Create API



## Home page

Lorem ipsum dolor sit amet consectetur adipisicing elit. Tenetur eos dolorem vero vel sequi voluptatibus assumenda ea libero cupiditate itaque, totam est rem possimus alias obcaecati omnis veniam! Corporis, nisi?

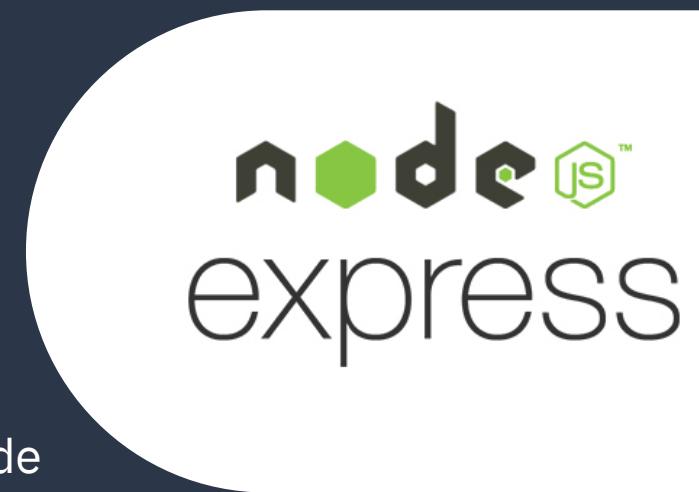
- [Users](#)
- [Posts](#) 

localhost:3000/api/posts

# Section 6

## สร้างเว็บเซิฟเวอร์ด้วย Express

- เริ่มต้นสร้างโปรเจกต์ใหม่และติดตั้ง Express
- ทำความรู้จักเกี่ยวกับ Middleware
- กบกวนเกี่ยวกับ REST API
- การใช้งาน Express เบื้องต้น
- กำหนด Middleware โดยใช้ Method use
- เปลี่ยน Port อัตโนมัติตามหมายเลข port ใน Process
- การส่งค่าพารามิเตอร์ไปพร้อมกับ HTTP Request
- การส่งค่าพารามิเตอร์หลายค่า
- ตัวอย่างการส่งค่าพารามิเตอร์ไปกับ url
- การใช้งาน Query String ในแอพพลิเคชัน
- การทดสอบส่ง HTTP Request ด้วย Postman หรือ Thunder Client ใน VS Code
- ติดตั้ง Module body-parser
- การเพิ่มข้อมูลใหม่ผ่าน Method POST
- อัปเดตข้อมูลด้วย Method PUT
- ลบข้อมูลด้วย Method DELETE
- การใช้งาน Nodemon รันแอพพลิเคชัน



What is Express ?

Routes

Request & Response

Middleware

Template Engines

Express

What is Express ?

Routes

Request & Response

Middleware

Template Engines

Express

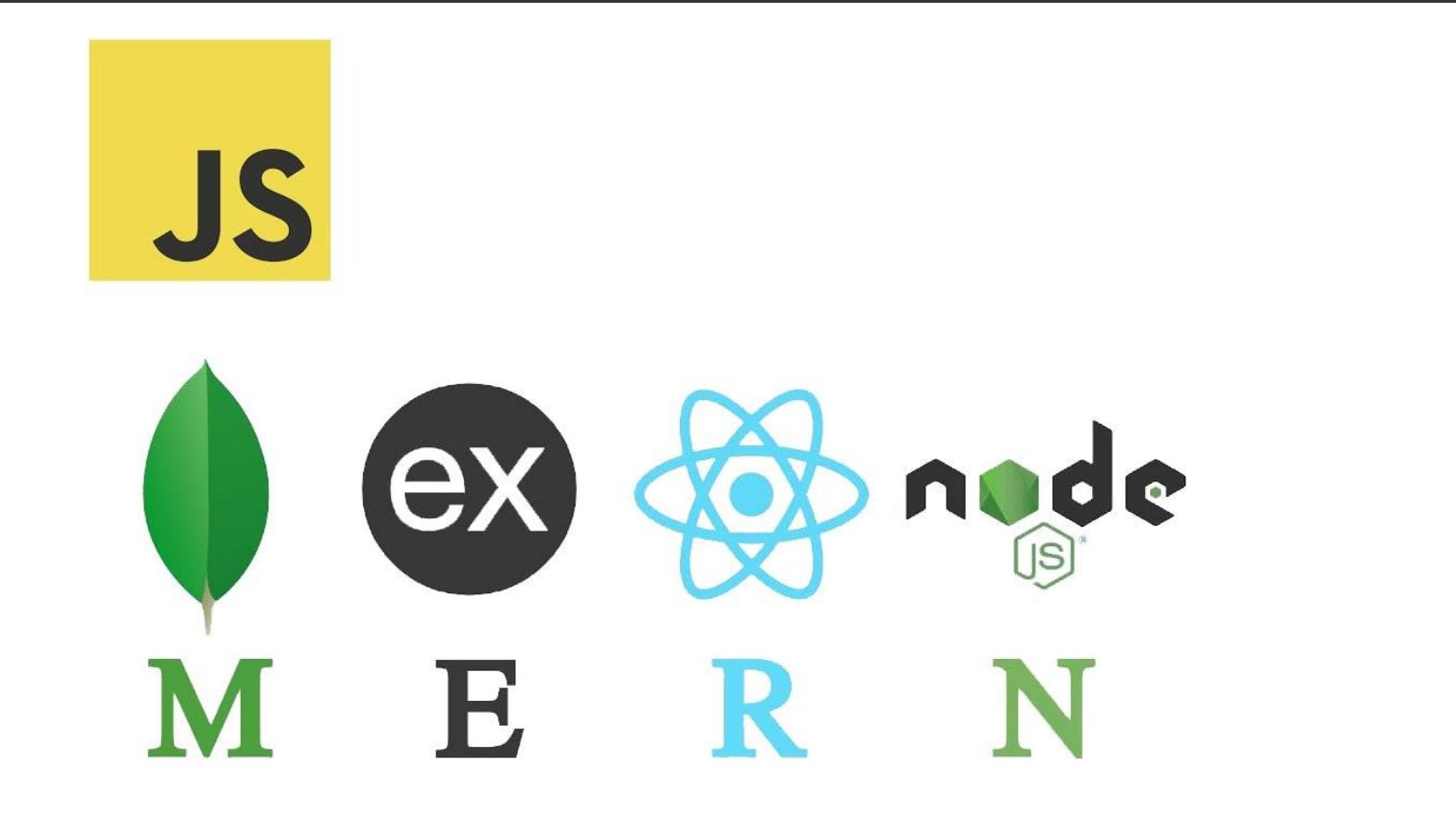
# Express

## What is Express ?

Express เป็นเฟรมเวอร์คยอดนิยมบน Node.js ที่ช่วยให้การสร้างเว็บเชิฟเวอร์สามารถทำได้ง่ายและรวดเร็ว

Express มีจุดเด่นเรื่องขนาดเล็ก ทำงานเร็ว เหมาะสำหรับนำมาสร้างเป็นเว็บ API เพื่อทำงานร่วมกับ Frontend อย่าง React, Angular และ Vue.js ได้เป็นอย่างดี

# Stack with Express (MERN)



# Stack with Express (MEAN)



# Stack with Express (MEVN)



# Why Express ?

Express

Lightweight

Simple

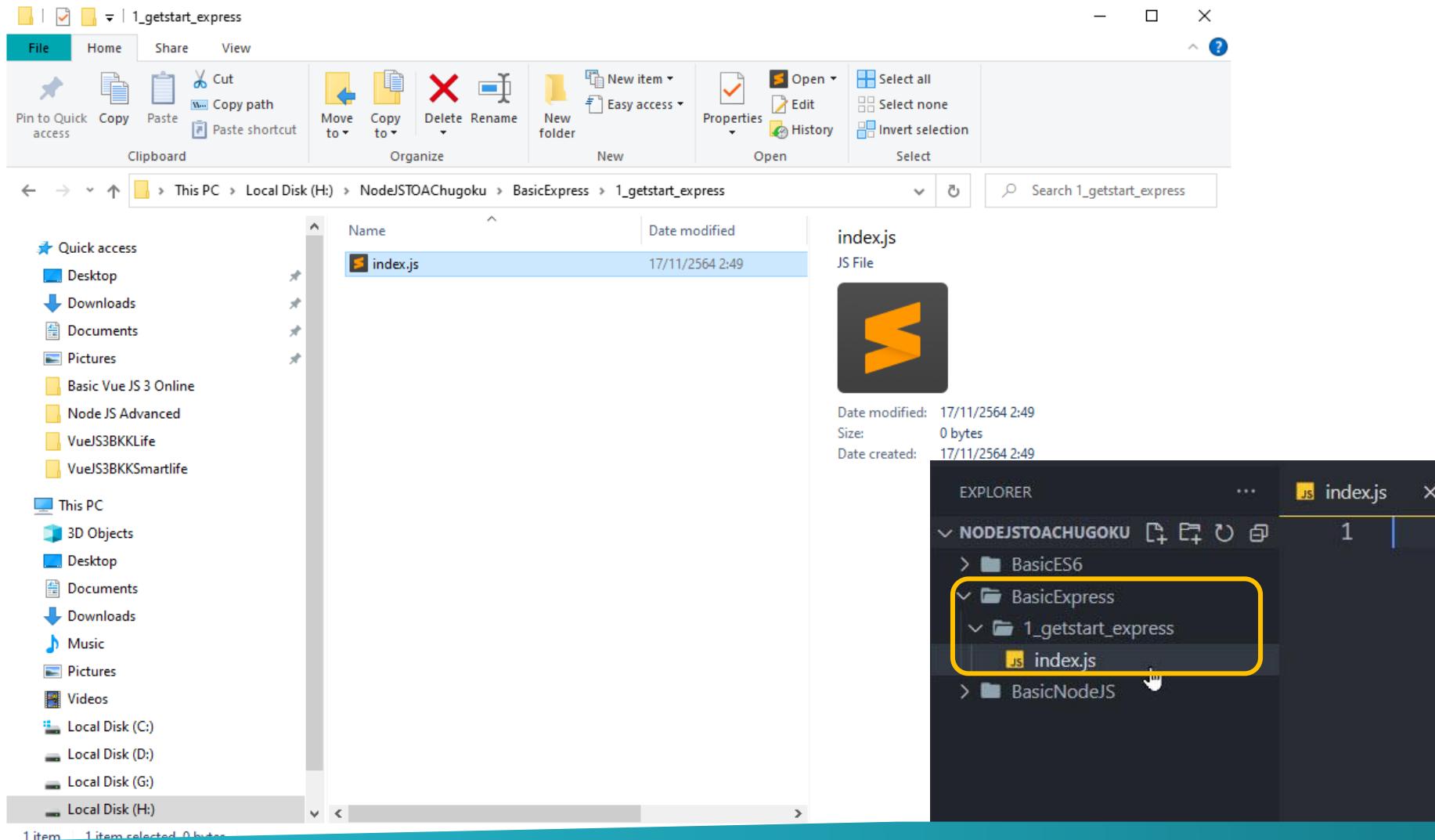
Extremely Fast

Unopinionated

JavaScript

Microservices

# เริ่มใช้งาน Express



# Install Express

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

H:\NodeJSTOAChugoku\BasicExpress\1_getstart_express>npm init -y
Wrote to H:\NodeJSTOAChugoku\BasicExpress\1_getstart_express\package.json:

{
  "name": "1_getstart_express",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \\\"Error: no test specified\\\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}
```

# Install Express

The screenshot shows a Visual Studio Code interface. The Explorer sidebar on the left lists a project structure under 'NODEJSTOACHUGOKU'. The 'package.json' file is open in the main editor area. The code is as follows:

```
1 {  
2   "name": "1_getstart_express",  
3   "version": "1.0.0",  
4   "description": "",  
5   "main": "index.js",  
6   "scripts": {  
7     "test": "echo \\"Error: no test specified\\" && exit 1"  
8   },  
9   "keywords": [],  
10  "author": "",  
11  "license": "ISC",  
12  "dependencies": {  
13    "express": "^4.17.1"  
14  }  
15}  
16
```

The 'express' dependency in the 'dependencies' section is highlighted with a yellow box. Below the editor, the terminal window shows the command 'npm install express' being run and its output:

```
H:\NodeJSTOACHugoku\BasicExpress\1_getstart_express>npm install express  
npm notice created a lockfile as package-lock.json. You should commit this file.  
npm WARN 1_getstart_express@1.0.0 No description  
npm WARN 1_getstart_express@1.0.0 No repository field.  
+ express@4.17.1  
added 50 packages from 37 contributors and audited 50 packages in 2.071s  
found 0 vulnerabilities
```

# Install Nodemon

The screenshot shows a VS Code interface with the following details:

- Explorer View:** Shows a project structure under "NODEJS TO A CHUGOKU" with folders like BasicES6, BasicExpress, and 1\_getstart\_express, along with their respective index.js files, node\_modules, package-lock.json, and package.json.
- Code Editor:** The "package.json" file is open, displaying its contents. Two specific sections are highlighted with yellow boxes:
  - The "scripts" section, which contains the "start" script: "start": "nodemon index.js",
  - The "dependencies" section, which lists "nodemon": "^2.0.15".
- Terminal:** The terminal window shows the command "npm install -D nodemon" being run in the directory H:\NodeJSTOAChugoku\BasicExpress\1\_getstart\_express. The output of the command is displayed below, showing the installation process and npm warnings.

```
H:\NodeJSTOAChugoku\BasicExpress\1_getstart_express>npm install -D nodemon
> nodemon@2.0.15 postinstall H:\NodeJSTOAChugoku\BasicExpress\1_getstart_express\node_modules\nodemon
> node bin/postinstall || exit 0

npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@~2.3.2 (node_modules\chokidar\node_modules\fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@2.3.2: wanted {"os":"darwin","arch":"any"} (current: {"os":"win32","arch":"x64"})
npm WARN 1_getstart_express@1.0.0 No description
npm WARN 1_getstart_express@1.0.0 No repository field.

+ nodemon@2.0.15
added 116 packages from 53 contributors and audited 167 packages in 15.531s
```

# Create Express Server

```
// Import Express
const express = require('express')

// Create express object
const app = express()

// Listen port 3000
app.listen(3000, ()=>{
  console.log('Listening to port 3000');
})
```

The screenshot shows a terminal window and a browser window side-by-side.

**Terminal:**

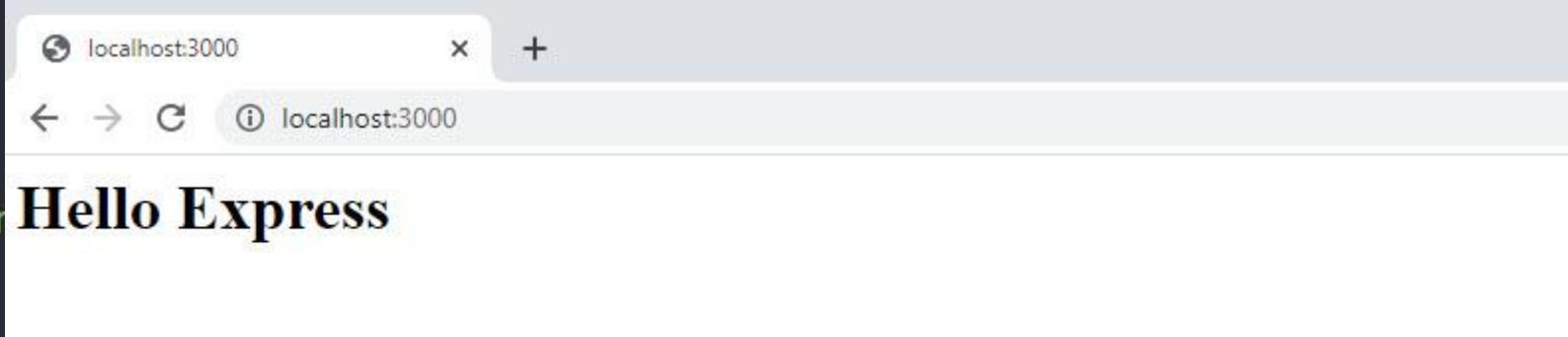
```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
H:\NodeJSTOAChugoku\BasicExpress\1_getstart_express>npm start
> 1_getstart_express@1.0.0 start H:\NodeJSTOAChugoku\BasicExpress
> nodemon index.js

[nodemon] 2.0.15
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node index.js`
```

**Browser:**

An Error dialog box is displayed, showing the URL `localhost:3000` and the message `Cannot GET /`.

# เพิ่ม method GET



```
index.js  x
1 // Import Express
2 const express = require('express')
3
4 // Create express object
5 const app = express()
6
7 // Create Method GET
8 app.get('/', (req, res)=>{
9     res.send('<h1>Hello Express</h1>')
10})
11
12 // Listen port 3000
13 app.listen(3000, ()=>{
14     console.log('Listening')
15})
```

The screenshot shows a code editor with a file named index.js containing Node.js code. The code imports Express, creates an express object, and defines a GET route for the root path ('/') that sends the string '<h1>Hello Express</h1>' as a response. It also listens on port 3000 and logs 'Listening' to the console. To the right of the code editor is a terminal window titled 'localhost:3000' showing the output of the application. The terminal window displays the text 'Hello Express'.

# What is Express ?

Routes

Request & Response

Middleware

Template Engines

# Express

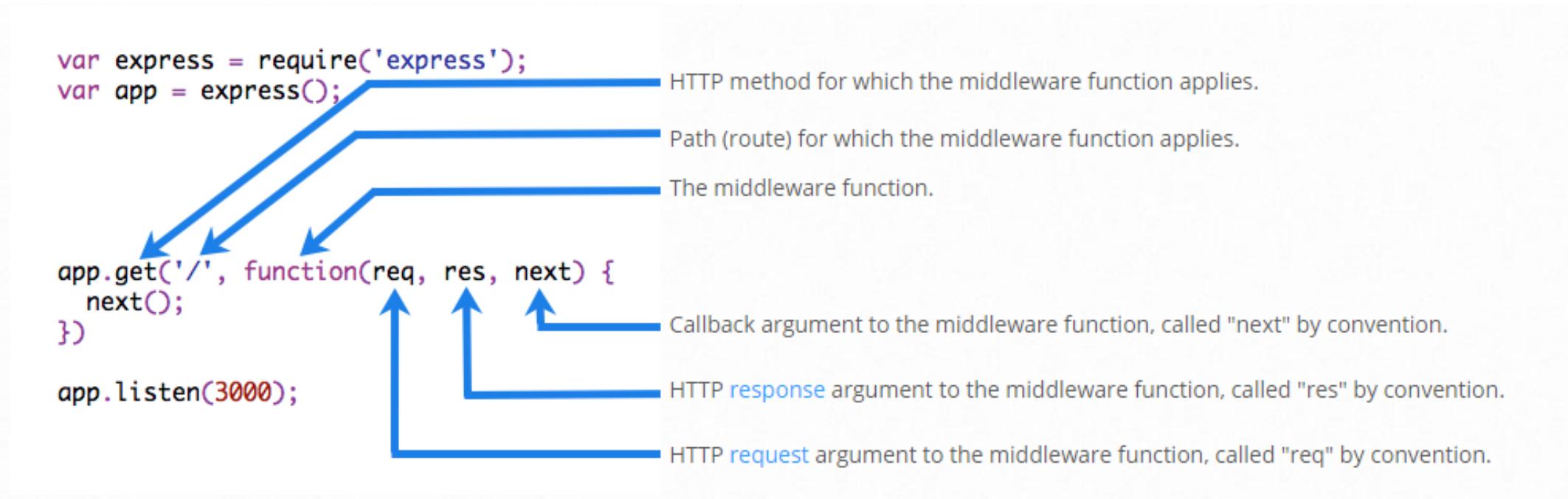
# การทำงานกับ Routing ใน Express

- การทำงานกับ Navigation
- การใช้งาน Routing
- การเขียน Router functions
- การประยุกต์ใช้ Routing แบบต่างๆ ในงานจริง



# Routes

เป็น Built-in URL routing system ใช้ในการบอกว่าต้องการตอบกลับ Request ในรูปแบบใด เช่นแสดงหน้าไฟล์ html หรือข้อมูลแบบ json



# Restful API



/apis/restaurants

POST

/apis/restaurants

GET

/apis/restaurants/1

GET

/apis/restaurants/1

PUT

/apis/restaurants/1

DELETE

# Create API method GET, POST

The screenshot shows a code editor with a file named index.js. The code defines an Express application with two routes: a GET endpoint at /api/restaurants and a POST endpoint at /api/restaurants. The GET endpoint returns the string 'GET API', and the POST endpoint returns 'POST API'. A yellow box highlights the route definitions for both methods. Below the code editor is a terminal window showing the results of running curl commands against the localhost API.

```
index.js  x
1 // Import Express
2 const express = require('express')
3
4 // Create express object
5 const app = express()
6
7 // Restful API
8 // Method GET
9 app.get('/api/restaurants',(req, res) => {
10   res.send('GET API')
11 }
12
13 // Method POST
14 app.post('/api/restaurants',(req, res) => {
15   res.send('POST API')
16 }
17
18 // Create Method GET
19 app.get('/', (req, res)=>{
20   res.send('<h1>Hello Express</h1>')

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
2: cmd +
```

```
H:\NodeJSTOAChugoku>curl -X GET http://localhost:3000/api/restaurants
GET API
H:\NodeJSTOAChugoku>curl -X POST http://localhost:3000/api/restaurants
POST API
H:\NodeJSTOAChugoku>
```

# Create API method PUT

The screenshot shows a code editor window for a file named 'index.js'. The code defines two API routes using the Express.js framework. The first route is a POST method for '/api/restaurants' which sends a response of 'POST API'. The second route is a PUT method for '/api/restaurants/:id' which sends a response of 'PUT API'. A yellow box highlights the PUT method code. Below the code editor, there are tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, and TERMINAL. In the TERMINAL tab, a command is being typed: 'curl -X PUT http://localhost:3000/api/restaurants/1'. The output of this command, 'PUT API', is also highlighted with a yellow box.

```
index.js
15     res.send('POST API')
16 }
17
18 // Method PUT
19 app.put('/api/restaurants/:id',(req, res) => {
20     res.send('PUT API')
21 }
22
23 // Create Method GET
24 app.get('/', (req, res)=>{
25     res.send('<h1>Hello Express</h1>')
26 }
27
```

```
H:\NodeJSTOACHugoku>curl -X PUT http://localhost:3000/api/restaurants/1
PUT API
H:\NodeJSTOACHugoku>
```

# Create API method DELETE

The screenshot shows a code editor window for a file named 'index.js'. The code defines several API routes using the Express.js framework:

```
19 app.put('/api/restaurants/:id', (req, res) => {
20   res.send('PUT API')
21 })
22
23 // Method DELETE
24 app.delete('/api/restaurants/:id', (req, res) => {
25   res.send('DELETE API')
26 })
27
28 // Create Method GET
29 app.get('/', (req, res) => {
30   res.send('<h1>Hello Express</h1>')
```

The code block from line 23 to 26 is highlighted with a yellow box. Below the editor, there is a terminal window showing the command and its output:

```
H:\NodeJSTOAChugoku>curl -X DELETE http://localhost:3000/api/restaurants/1
DELETE API
H:\NodeJSTOAChugoku>
```

# Create API method GET with Parameter

The screenshot shows a code editor with a dark theme. A yellow box highlights the code for retrieving a restaurant by ID. Below the editor is a terminal window showing the command and its execution.

```
index.js
6
7 // Restful API
8 // Method GET
9 app.get('/api/restaurants',(req, res) => {
10   res.send('GET API')
11 })
12
13 // Method GET by ID
14 app.get('/api/restaurants/:id',(req, res) => {
15   res.send('GET ONE API')
16 })
17
18 // Method POST
19 app.post('/api/restaurants',(req, res) => {
20   res.send('POST API')
21 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
H:\NodeJSTOAChugoku>curl -X GET http://localhost:3000/api/restaurants/1
GET ONE API
H:\NodeJSTOAChugoku>
```

# Create API Route file

The screenshot shows a Visual Studio Code interface with the following details:

- File Explorer (Left):** Shows a project structure under "NODEJSSTOACHUGOKU". The "routes" folder contains a file named "restaurants.js", which is highlighted with a yellow box.
- Code Editor (Right):** The file "restaurants.js" is open, displaying the following code:

```
const express = require('express')
const router = express.Router()

// Restful API
// Method GET
router.get('/',(req, res) => {
    res.send('GET API')
})

// Method GET by ID
router.get('/:id',(req, res) => {
    res.send('GET ONE API')
})

// Method POST
router.post('/',(req, res) => {
    res.send('POST API')
})

// Method PUT
router.put('/:id',(req, res) => {
    res.send('PUT API')
})

// Method DELETE
router.delete('/:id',(req, res) => {
    res.send('DELETE API')
})
```

# Create API Route file

The screenshot shows a Visual Studio Code interface with the following details:

- File Menu:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Title Bar:** index.js - NodeJSTOACHUGOKU - Visual Studio Code.
- Explorer:** Shows a project structure under NODEJSTOACHUGOKU:
  - BasicES6
  - BasicExpress
    - 1\_getstart\_express
      - node\_modules
    - routes
      - restaurants.js
      - index.js (highlighted with a yellow border)
  - package-lock.json
  - package.json
  - BasicNodeJS
- Code Editor:** The file index.js is open, showing the following code:

```
// Import Express
const express = require('express')

// Import Router
const restaurantsRouter = require('./routes/restaurants')

// Create express object
const app = express()

// Routes
app.use('/api/restaurants', restaurantsRouter)

// Create Method GET
app.get('/', (req, res)=>{
    res.send('<h1>Hello Express</h1>')
})

// Listen port 3000
app.listen(3000, ()=>{
    console.log('Listening to port 3000');
})
```

The code is highlighted with syntax coloring, and two specific sections are enclosed in yellow boxes:
  - `// Import Router` and `const restaurantsRouter = require('./routes/restaurants')`
  - `app.use('/api/restaurants', restaurantsRouter)`

What is Express ?

Routes

Request & Response

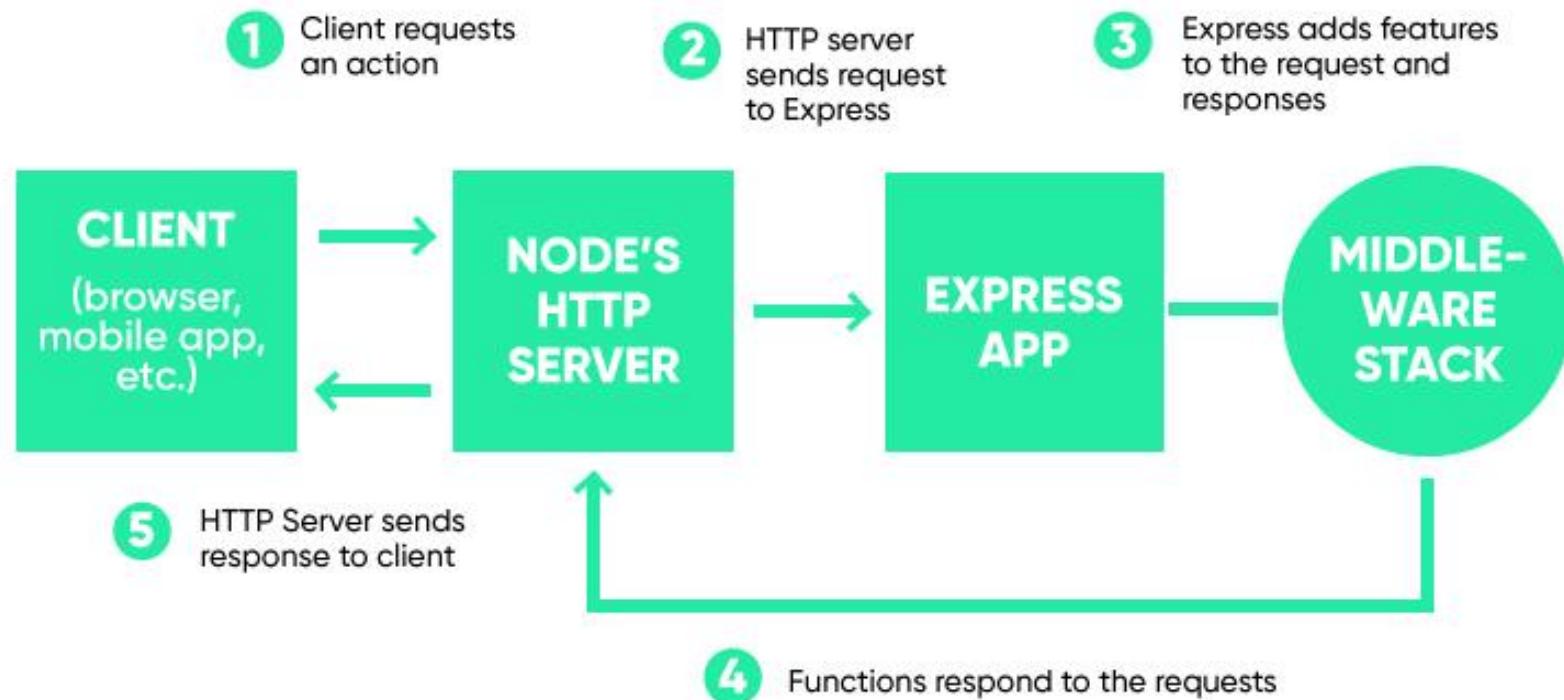
Middleware

Template Engines

Express

# Request & Responses

ทุก Request ที่ส่งเข้าไปใน Express Server ประกอบไปด้วยรอบเจ็กต์ Request และ Response โดย Request คือข้อมูลต่างๆ ที่ส่งไปหา Server และ Response ก็จะเป็น Method ต่าง ๆ ที่ใช้ส่งกลับไปหา Client



# Request & Responses

## Request

req.params

req.query

req.body

## Response

res.send()

res.sendFile()

res.sendStatus()

res.json()

# การใช้งาน Template Engine ใน Express

- รู้จักกับ Template Engine
- รู้จัก Jade / Pug Template engine และการใช้งานเบื้องต้น
- การแสดงผลและการใช้งานตัวแปรต่างๆ ใน Template
- การกำหนดเงื่อนไข (condition) ใน Template
- การทำซ้ำ (loop) ใน Template
- การแปลงโค้ด HTML เป็น Pug
- รู้จักกับ EJS Template Engine และการใช้งานเบื้องต้น
- แทรกโค้ด JavaScript ลงใน EJS Template



# Node.JS กับการทำงานกับฐานข้อมูล

- ติดตั้งฐานข้อมูลที่ต้องการเช่น MySQL, MongoDB หรือ MS SQL Server
- ติดตั้ง Package สำหรับเชื่อมต่อกับฐานข้อมูลที่ใช้
- สร้างฐานข้อมูลใน MS SQL Server
- สร้างโปรเจ็คต์ Node.js ใหม่สำหรับเชื่อมต่อกับ MS SQL Server
- สร้างไฟล์ .env สำหรับ config ตัวแปรในการเชื่อมต่อฐานข้อมูล
- เขียน Node.js API กับ MS SQL Server
- สร้าง SQL Data Access Layer
- สร้าง Database Client Plugin



# Create Database in MS SQL Server

```
CREATE DATABASE smartinvdb
GO

USE smartinvdb
GO

SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[products](
    [ProductID] [int] IDENTITY(1,1) NOT NULL,
    [CategoryID] [int] NULL,
    [ProductName] [nvarchar](50) NULL,
    [UnitPrice] [decimal](18, 0) NULL,
    [ProductPicture] [varchar](1024) NULL,
    [UnitInStock] [int] NULL,
    [CreatedDate] [datetime] NULL,
    [ModifiedDate] [datetime] NULL,
PRIMARY KEY CLUSTERED
(
    [ProductID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
)

GO
```

# Create Database in MS SQL Server

The screenshot shows the Microsoft SQL Server Object Explorer on the left and the SQL Server Object Explorer on the right.

**Object Explorer (Left):**

- smartinvdb
  - Database Diagrams
  - Tables
    - System Tables
    - FileTables
    - External Tables
    - Graph Tables
  - dbo.products
    - Columns
      - ProductID (PK, int, not null)
      - CategoryID (int, null)
      - ProductName (nvarchar(50), null)
      - UnitPrice (decimal(18,0), null)
      - ProductPicture (varchar(1024), null)
      - UnitInStock (int, null)
      - CreatedDate (datetime, null)
      - ModifiedDate (datetime, null)
    - Keys
    - Constraints
    - Triggers
    - Indexes
    - Statistics
  - Views
  - External Resources
  - Synonyms
  - Programmability
  - Service Broker
  - Storage
  - Security

**SQL Server Object Explorer (Right):**

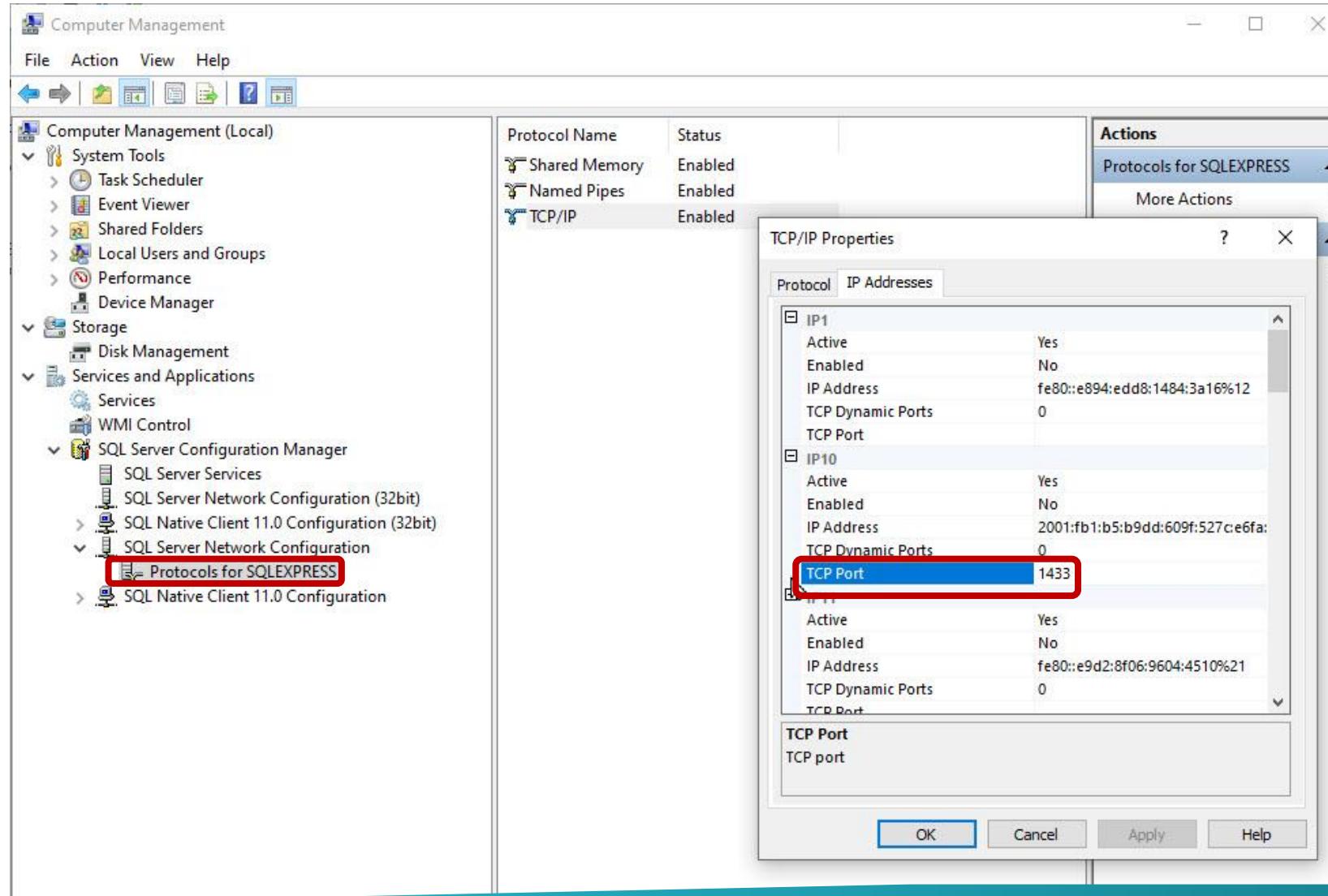
SAMITPCGENIUS\SQL2019 - db - dbo.products

Column Name	Data Type	Allow Nulls
ProductID	int	<input type="checkbox"/>
CategoryID	int	<input checked="" type="checkbox"/>
ProductName	nvarchar(50)	<input checked="" type="checkbox"/>
UnitPrice	decimal(18, 0)	<input checked="" type="checkbox"/>
ProductPicture	varchar(1024)	<input checked="" type="checkbox"/>
UnitInStock	int	<input checked="" type="checkbox"/>
CreatedDate	datetime	<input checked="" type="checkbox"/>
ModifiedDate	datetime	<input checked="" type="checkbox"/>
		<input type="checkbox"/>

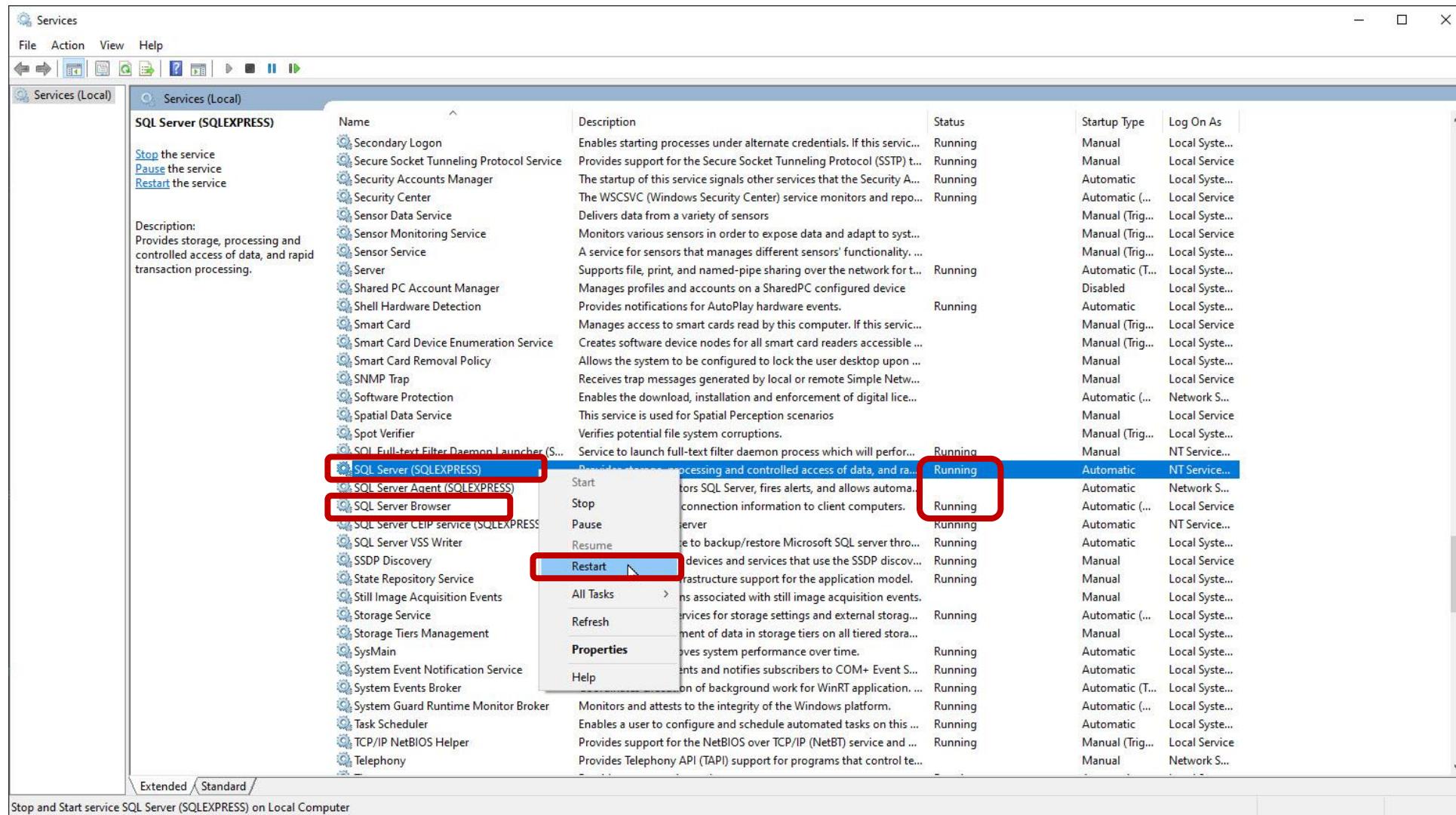
# Insert Demo data to table products

SAMITPCGENIUS\SQ...db - dbo.products								
	ProductID	CategoryID	ProductName	UnitPrice	ProductPicture	UnitInStock	CreatedDate	ModifiedDate
	1	1	iPhone 13 Pro ...	55000	https://www.mxphone.com/wp-content/uploads... 5		2021-11-22 00:00:00.000	2021-11-22 00:00:00.000
	2	1	iPad Pro 2021	18500	https://cdn.siamphone.com/spec/apple/images/... 10		2021-11-20 00:00:00.000	2021-11-20 00:00:00.000
▶*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

# Setting TCP/IP MS SQL Server



# Restart SQL Server Services and Enable running SQL Server Browser

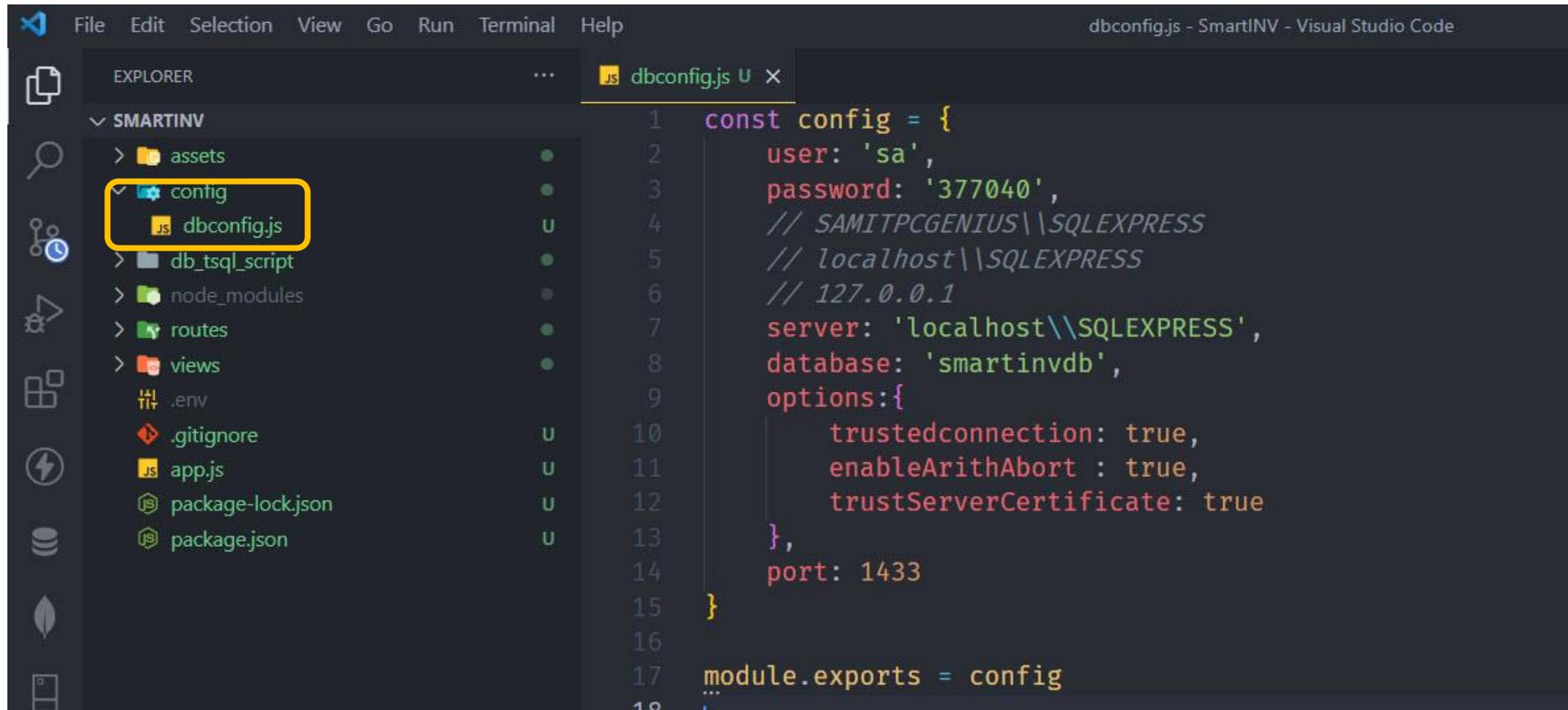


# Install node mssql Package

```
npm install mssql
```

```
16  "dependencies": {  
17    "ejs": "^3.1.6",  
18    "express": "^4.17.1",  
19    "express-ejs-layouts": "^2.5.1",  
20    "mssql": "^7.3.0"  
21  }
```

# Create dbconfig.js for mssql



```
const config = {
    user: 'sa',
    password: '377040',
    // SAMITPCGENIUS\SQLEXPRESS
    // localhost\SQLEXPRESS
    // 127.0.0.1
    server: 'localhost\\SQLEXPRESS',
    database: 'smartinvdb',
    options: {
        trustedconnection: true,
        enableArithAbort : true,
        trustServerCertificate: true
    },
    port: 1433
}
module.exports = config
```

# Connect mssql in app.js

The screenshot shows the Visual Studio Code interface with the file 'app.js' open. The code is written in JavaScript and uses the Express framework. It includes imports for 'express', 'mssql', and 'express-ejs-layouts'. The code then connects to a database using the 'mssql' module and logs a message if successful. The code is annotated with two yellow boxes highlighting the database connection logic.

```
// Import Express
const express = require('express')

//Use mssql database
const sql = require('mssql');
const dbConfig = require('./config/dbconfig')

// Import EJS Layout
const expressLayouts = require('express-ejs-layouts')

// Import file frontend.js
const frontendRouter = require('./routes/frontend')
// Import file backend.js
const backendRouter = require('./routes/backend')

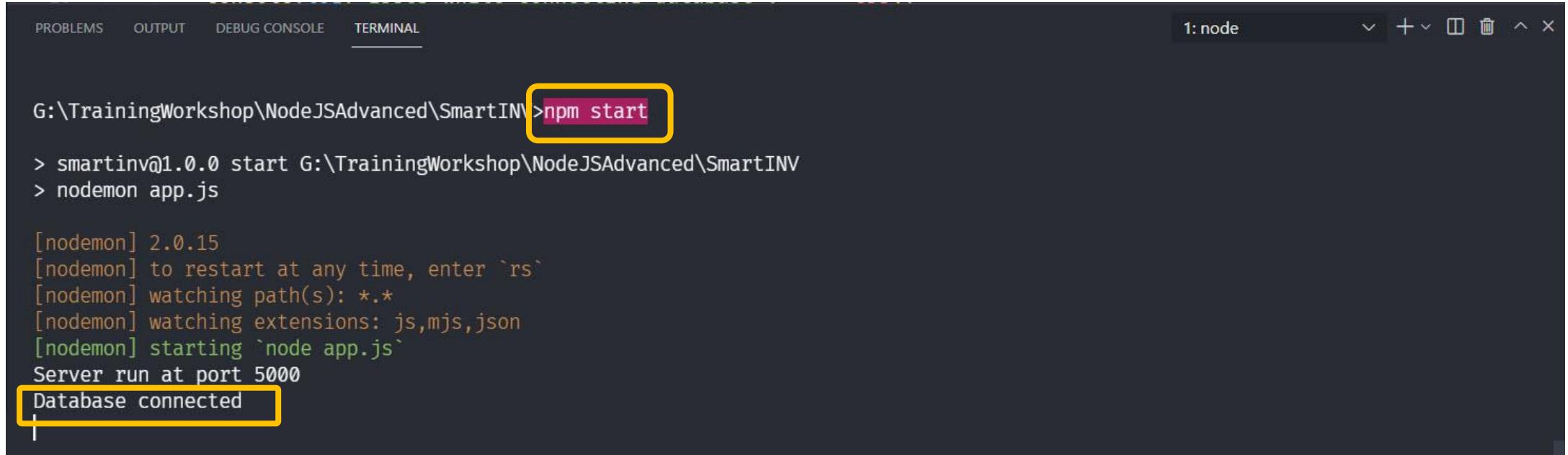
// MSSQL Connect
sql.connect(dbConfig, function (err) {
  if (err) {
    console.log("Error while connecting database :- " + err);
  }
  else {
    console.log("Database connected");
  }
});

// Create express object
const app = express()

// กำหนด Folder สำหรับอัตว์ express ร่าไฟล์ css , images อู่ path นน
app.use(express.static('assets'))

// กำหนด Template Engine
app.use(expressLayouts)
app.set('layout','./layouts/frontend')
```

# Test run connect

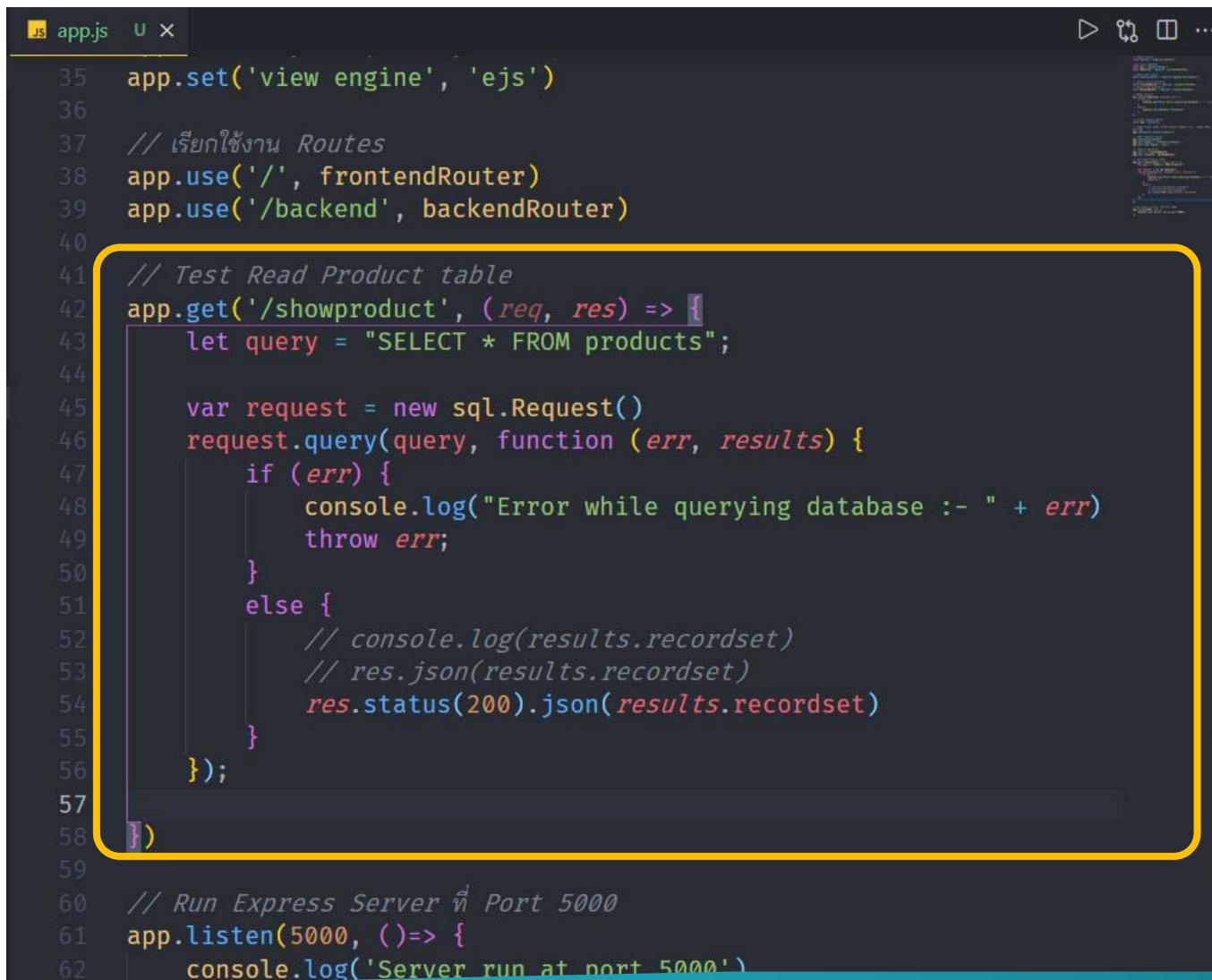


The screenshot shows a terminal window in a dark-themed IDE interface. The tabs at the top are PROBLEMS, OUTPUT, DEBUG CONSOLE, and TERMINAL, with TERMINAL being the active tab. The title bar indicates the session is titled '1: node'. The command 'npm start' is highlighted with a yellow box. The output shows the execution of the command and the logs from nodemon starting the application. The message 'Database connected' is also highlighted with a yellow box.

```
G:\TrainingWorkshop\NodeJSAdvanced\SmartINV>npm start
> smartinv@1.0.0 start G:\TrainingWorkshop\NodeJSAdvanced\SmartINV
> nodemon app.js

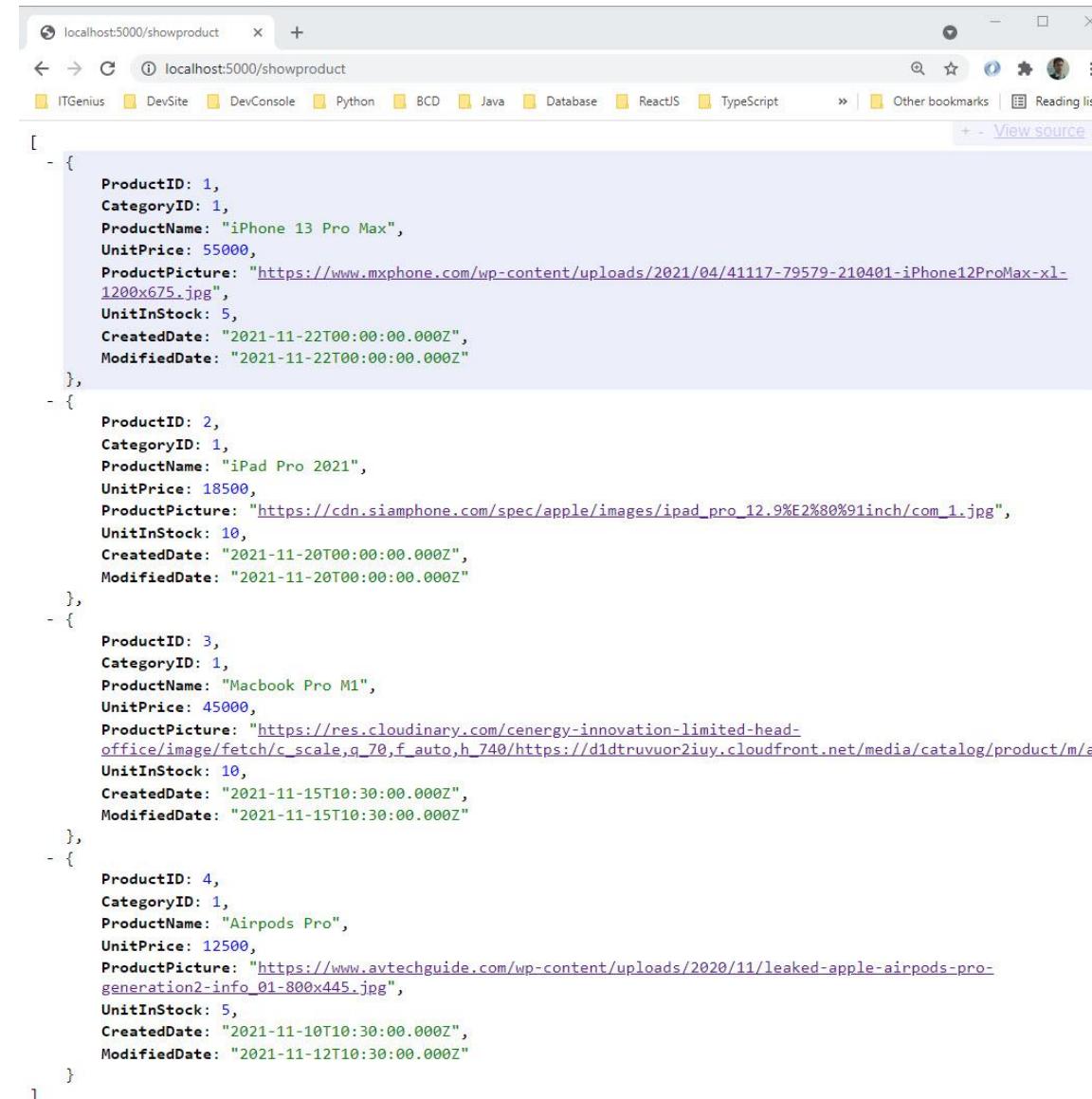
[nodemon] 2.0.15
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node app.js`
Server run at port 5000
Database connected
```

# Create Function to connect to database and execute query



```
js app.js  ✘ ...  
35 app.set('view engine', 'ejs')  
36  
37 // เรียกใช้งาน Routes  
38 app.use('/', frontendRouter)  
39 app.use('/backend', backendRouter)  
40  
41 // Test Read Product table  
42 app.get('/showproduct', (req, res) => {  
43     let query = "SELECT * FROM products";  
44  
45     var request = new sql.Request()  
46     request.query(query, function (err, results) {  
47         if (err) {  
48             console.log("Error while querying database :- " + err)  
49             throw err;  
50         }  
51         else {  
52             // console.log(results.recordset)  
53             // res.json(results.recordset)  
54             res.status(200).json(results.recordset)  
55         }  
56     });  
57 }  
58  
59 // Run Express Server ที่ Port 5000  
60 app.listen(5000, ()=> {  
61     console.log('Server run at port 5000')  
62 }
```

# Result



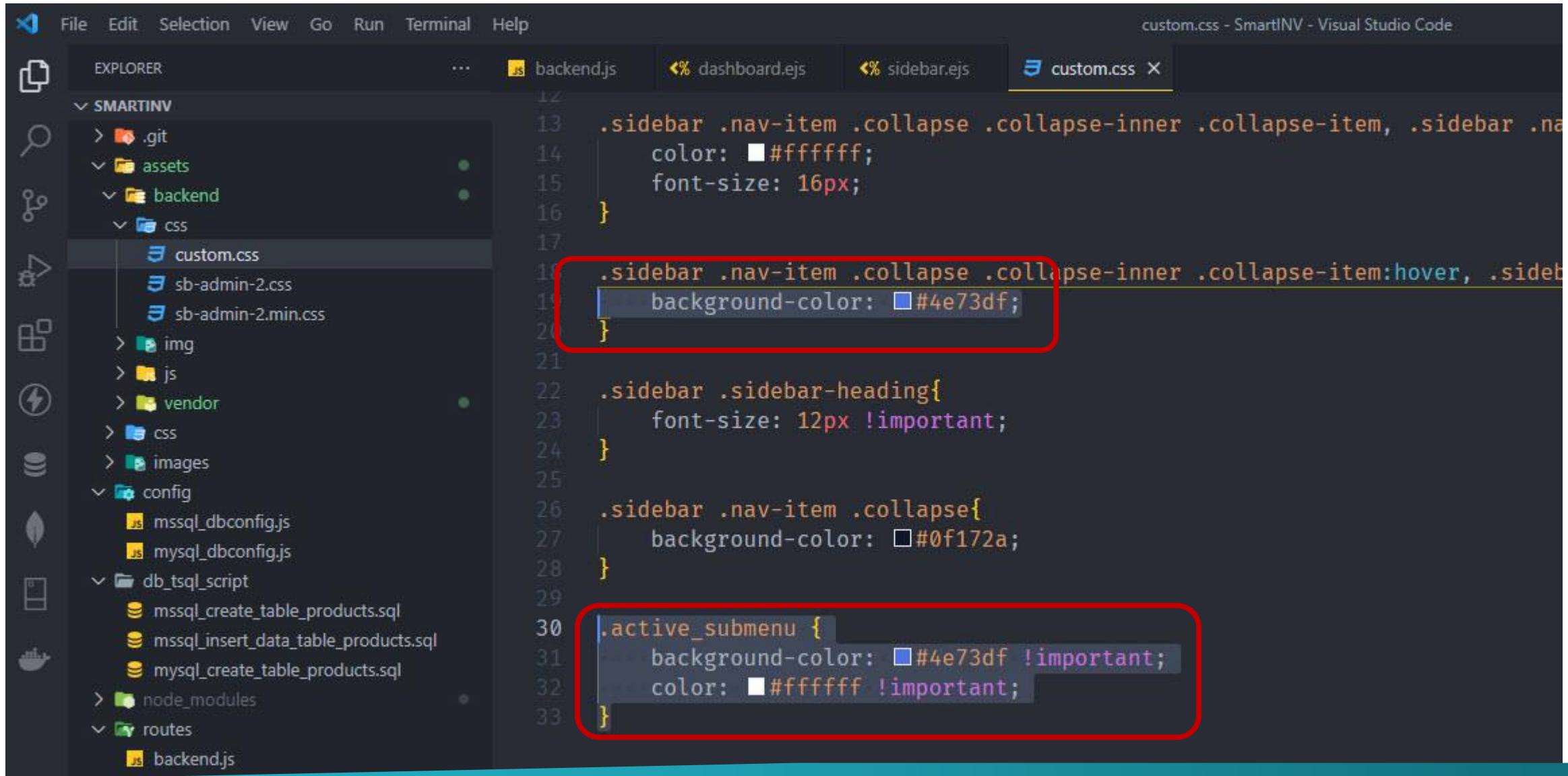
A screenshot of a web browser window titled "localhost:5000/showproduct". The page displays a JSON array of product data. Each product object contains the following fields: ProductID, CategoryID, ProductName, UnitPrice, ProductPicture, UnitInStock, CreatedDate, and ModifiedDate. The browser's address bar shows the URL "localhost:5000/showproduct". The top navigation bar includes links for ITGenius, DevSite, DevConsole, Python, BCD, Java, Database, ReactJS, TypeScript, and others.

```
[{"ProductID": 1, "CategoryID": 1, "ProductName": "iPhone 13 Pro Max", "UnitPrice": 55000, "ProductPicture": "https://www.mxphone.com/wp-content/uploads/2021/04/41117-79579-210401-iPhone12ProMax-xl-1200x675.jpg", "UnitInStock": 5, "CreatedDate": "2021-11-22T00:00:00.000Z", "ModifiedDate": "2021-11-22T00:00:00.000Z"}, {"ProductID": 2, "CategoryID": 1, "ProductName": "iPad Pro 2021", "UnitPrice": 18500, "ProductPicture": "https://cdn.siamphone.com/spec/apple/images/ipad_pro_12.9%E2%80%91inch/com_1.jpg", "UnitInStock": 10, "CreatedDate": "2021-11-20T00:00:00.000Z", "ModifiedDate": "2021-11-20T00:00:00.000Z"}, {"ProductID": 3, "CategoryID": 1, "ProductName": "Macbook Pro M1", "UnitPrice": 45000, "ProductPicture": "https://res.cloudinary.com/cenergy-innovation-limited-head-office/image/fetch/c_scale,q_70,f_auto,h_740/https://d1dtruvor2iuy.cloudfront.net/media/catalog/product/m/a/m", "UnitInStock": 10, "CreatedDate": "2021-11-15T10:30:00.000Z", "ModifiedDate": "2021-11-15T10:30:00.000Z"}, {"ProductID": 4, "CategoryID": 1, "ProductName": "Airpods Pro", "UnitPrice": 12500, "ProductPicture": "https://www.avtechguide.com/wp-content/uploads/2020/11/leaked-apple-airpods-pro-generation2-info_01-800x445.jpg", "UnitInStock": 5, "CreatedDate": "2021-11-10T10:30:00.000Z", "ModifiedDate": "2021-11-12T10:30:00.000Z"}]
```

<https://github.com/une6/nodejs-mssql-simple-crud/blob/master/index.js>

# **Config Active Menu**

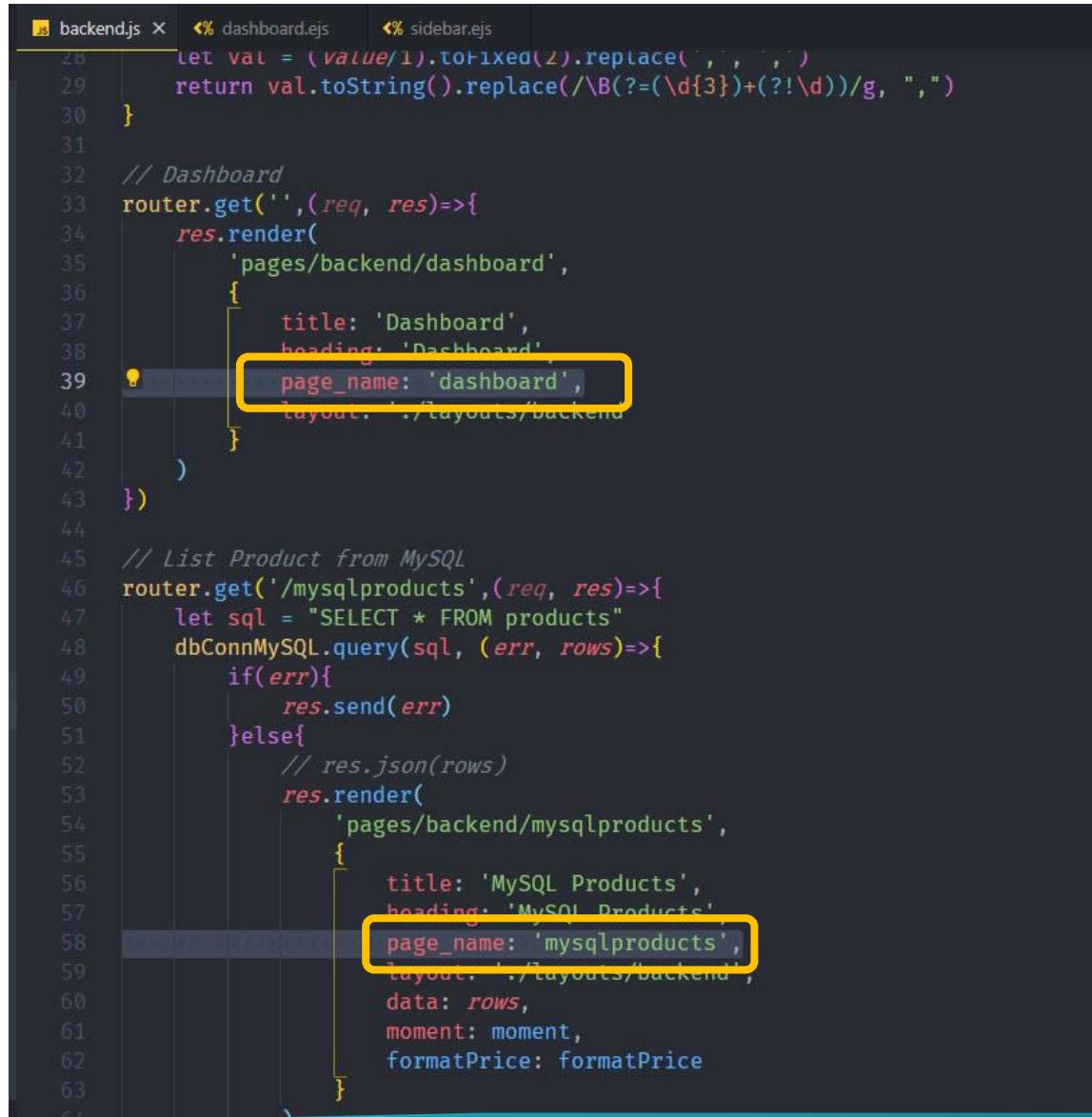
# Config Active Menu



The screenshot shows the Visual Studio Code interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Title Bar:** custom.css - SmartINV - Visual Studio Code.
- Explorer:** Shows the project structure under SMARTINV:
  - .git
  - assets
    - backend
      - css (containing custom.css)
  - img
  - js
  - vendor
  - css
  - images
  - config
    - mssql\_dbconfig.js
    - mysql\_dbconfig.js
  - db\_tsqL\_script
    - mssql\_create\_table\_products.sql
    - mssql\_insert\_data\_table\_products.sql
    - mysql\_create\_table\_products.sql
  - node\_modules
  - routes
  - backend.js
- Editor:** The custom.css file is open, showing CSS code for sidebar navigation items. Two specific rules are highlighted with red boxes:
  - `.sidebar .nav-item .collapse .collapse-inner .collapse-item:active, .sidebar .nav-item .collapse .collapse-inner .collapse-item:hover { background-color: #4e73df; }`
  - `.active_submenu { background-color: #4e73df !important; color: #ffffff !important; }`

# Config Active Menu



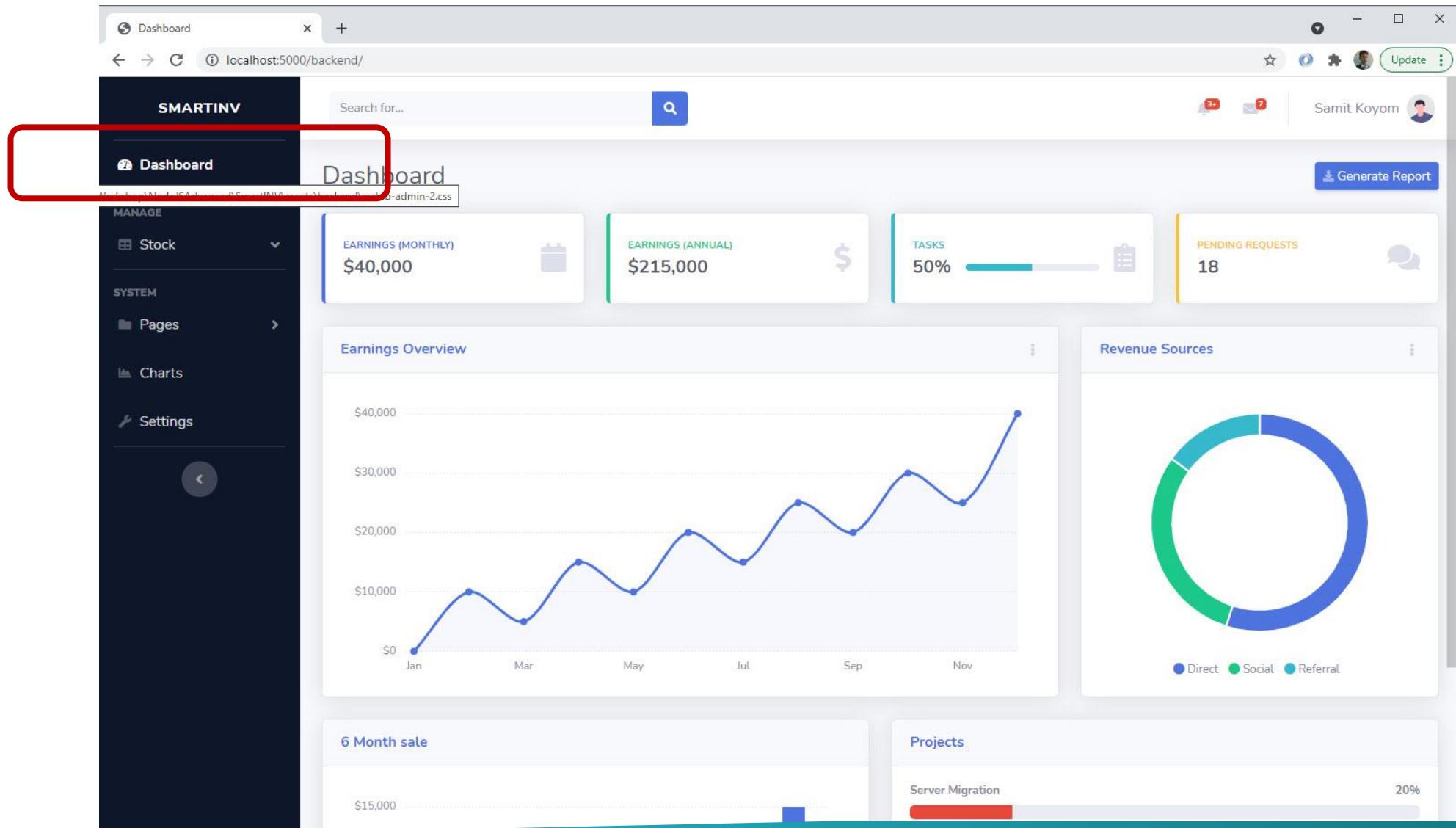
```
backend.js  dashboard.ejs  sidebar.ejs
28     let val = (value/1).toFixed(2).replace(',', '.', '.')
29     return val.toString().replace(/(\B(?=(\d{3})+(\?!d))/g, ",")
30   }
31
32 // Dashboard
33 router.get('/', (req, res) =>{
34   res.render(
35     'pages/backend/dashboard',
36     {
37       title: 'Dashboard',
38       heading: 'Dashboard',
39       page_name: 'dashboard',
40       layout: './layouts/backend'
41     }
42   )
43 })
44
45 // List Product from MySQL
46 router.get('/mysqlproducts', (req, res) =>{
47   let sql = "SELECT * FROM products"
48   dbConnMySQL.query(sql, (err, rows) =>{
49     if(err){
50       res.send(err)
51     }else{
52       // res.json(rows)
53       res.render(
54         'pages/backend/mysqlproducts',
55         {
56           title: 'MySQL Products',
57           heading: 'MySQL Products',
58           page_name: 'mysqlproducts',
59           layout: './layouts/backend',
60           data: rows,
61           moment: moment,
62           formatPrice: formatPrice
63         }
64       )
65     }
66   })
67 })
```

# Config Active Menu

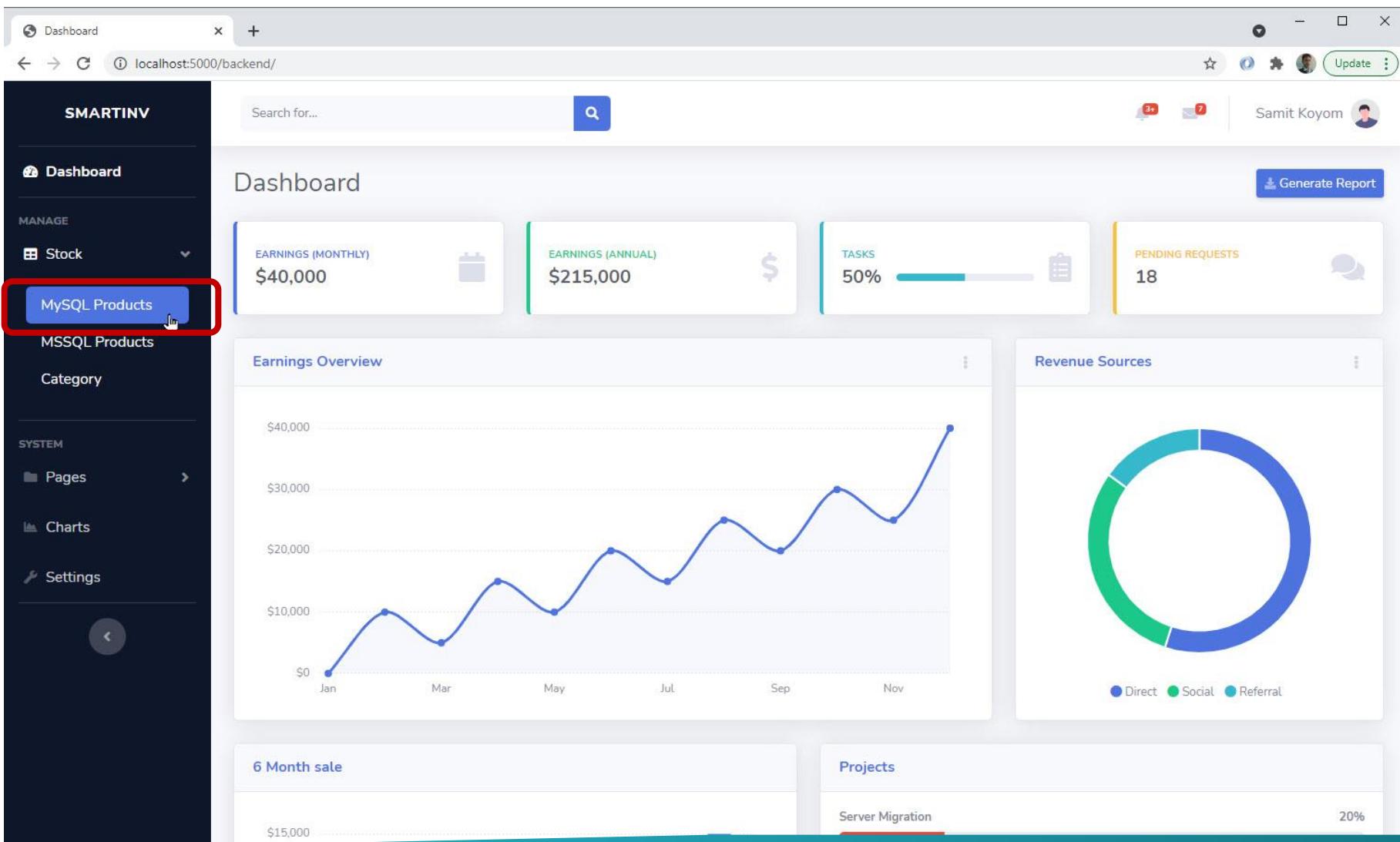
```
<% sidebar.ejs %>

17
18    <!-- Divider -->
19    <hr class="sidebar-divider">
20
21    <!-- Heading -->
22    <div class="sidebar-heading">
23        Manage
24    </div>
25
26    <!-- Nav Item - Pages Collapse Menu -->
27    <li class="nav-item <% if (page_name === 'mysqlproducts' || page_name === 'mssqlproducts' || page_name === 'category') { %>active<% } %>">
28        <a class="nav-link" href="#" data-toggle="collapse" data-target="#collapseTwo" aria-expanded="true" aria-controls="collapseTwo">
29            <i class="fas fa-fw fa-table"></i>
30            <span>Stock</span>
31        </a>
32        <div id="collapseTwo" class="collapse <% if (page_name === 'mysqlproducts' || page_name === 'mssqlproducts' || page_name === 'category') { %>show<% } %>" aria-labelledby="headingTwo" data-parent="#accordionSidebar">
33            <div class="py-2 collapse-inner rounded">
34                <a class="collapse-item <% if (page_name === 'mysqlproducts') { %>active_submenu<% } %>" href="/backend/mysqlproducts">MySQL Products</a>
35                <a class="collapse-item <% if (page_name === 'mssqlproducts') { %>active_submenu<% } %>" href="/backend/mssqlproducts">MSSQL Products</a>
36                <a class="collapse-item <% if (page_name === 'category') { %>active_submenu<% } %>" href="/backend/category">Category</a>
37            </div>
38        </div>
39    </li>
40
41    <!-- Divider -->
42    <hr class="sidebar-divider">
43
44    <!-- Heading -->
```

# Result



# Result



# Result

The screenshot shows a web application interface for managing MySQL products. The left sidebar has a dark theme with white text and icons. It includes sections for Dashboard, Manage (Stock, MySQL Products, MSSQL Products, Category), and System (Pages, Charts, Settings). The 'MySQL Products' button is highlighted with a red rectangle. The main content area has a light gray background. At the top, there's a search bar with placeholder text 'Search for...' and a magnifying glass icon. To the right of the search bar are three small circular icons with numbers (3+, 7, 1) and a user profile for 'Samit Koyom'. Below the search bar is a green button labeled '+ Create product'. The main table is titled 'MySQL Products' and has a blue header row with columns: Image, ID, Name, Price, Qty, Created, Updated, and Manage. There are four rows of data, each representing a product: iPhone 13 Pro Max (ID 1, Price 55,000.00), iPad Pro 2021 (ID 3, Price 18,500.00), Macbook Pro M1 (ID 4, Price 45,000.00), and Airpods Pro (ID 5, Price 12,500.00). Each row has a 'Manage' column with three buttons: 'View' (blue), 'Edit' (yellow), and 'Delete' (red).

Image	ID	Name	Price	Qty	Created	Updated	Manage
	1	iPhone 13 Pro Max	55,000.00	3	22 November 2021 23:56	22 November 2021 23:56	<button>View</button> <button>Edit</button> <button>Delete</button>
	3	iPad Pro 2021	18,500.00	10	22 November 2021 23:57	22 November 2021 23:57	<button>View</button> <button>Edit</button> <button>Delete</button>
	4	Macbook Pro M1	45,000.00	10	22 November 2021 23:57	22 November 2021 23:57	<button>View</button> <button>Edit</button> <button>Delete</button>
	5	Airpods Pro	12,500.00	5	22 November 2021 23:58	22 November 2021 23:58	<button>View</button> <button>Edit</button> <button>Delete</button>

# Result

The screenshot shows the SMARTINV application interface. On the left, a dark sidebar menu includes 'Dashboard', 'MANAGE' (with 'Stock' selected), 'MySQL Products', 'MSSQL Products' (which is highlighted with a red box), 'Category', 'SYSTEM' (with 'Pages' selected), 'Charts', and 'Settings'. The main content area is titled 'MSSQL Products' and displays a table of product data. The table has columns: Image, ID, Name, Price, Qty, Created, Updated, and Manage. There are four rows of data:

Image	ID	Name	Price	Qty	Created	Updated	Manage
	1	iPhone 13 Pro Max	55,000.00	5	22 November 2021 07:00	22 November 2021 07:00	<button>View</button> <button>Edit</button> <button>Delete</button>
	2	iPad Pro 2021	18,500.00	10	20 November 2021 07:00	20 November 2021 07:00	<button>View</button> <button>Edit</button> <button>Delete</button>
	3	Macbook Pro M1	45,000.00	10	15 November 2021 17:30	15 November 2021 17:30	<button>View</button> <button>Edit</button> <button>Delete</button>
	4	Airpods Pro	12,500.00	5	10 November 2021 17:30	12 November 2021 17:30	<button>View</button> <button>Edit</button> <button>Delete</button>

# Result

The screenshot shows the SMARTINV backend application interface. On the left, a dark sidebar menu lists several categories: Dashboard, Stock (selected), MySQL Products, MSSQL Products, and Category. The 'Category' item is highlighted with a red rectangle. The main content area is titled 'Category' and contains a search bar and a green '+ Create category' button. Below the title, there is a placeholder text: 'Lorem ipsum dolor, sit amet consectetur adipisicing elit. Reprehenderit tempora incidunt eligendi excepturi, nemo preferendis adipisci unde aperiam porro quo cumque. Dolorem cupiditate quos explicabo ad, eos autem tempore preferendis!'. At the top right of the main area, there are notifications for 3+ messages and 7 updates, along with a user profile for Samit Koyom.

# React and NodeJS with Docker Workshop Progress



Day 1



Day 2



Day 3



Day 4



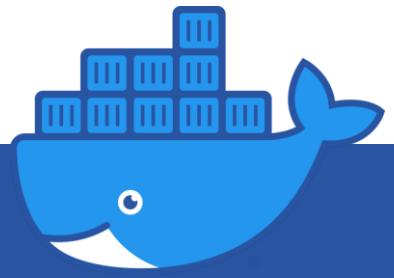
Day 5

# DAY 4

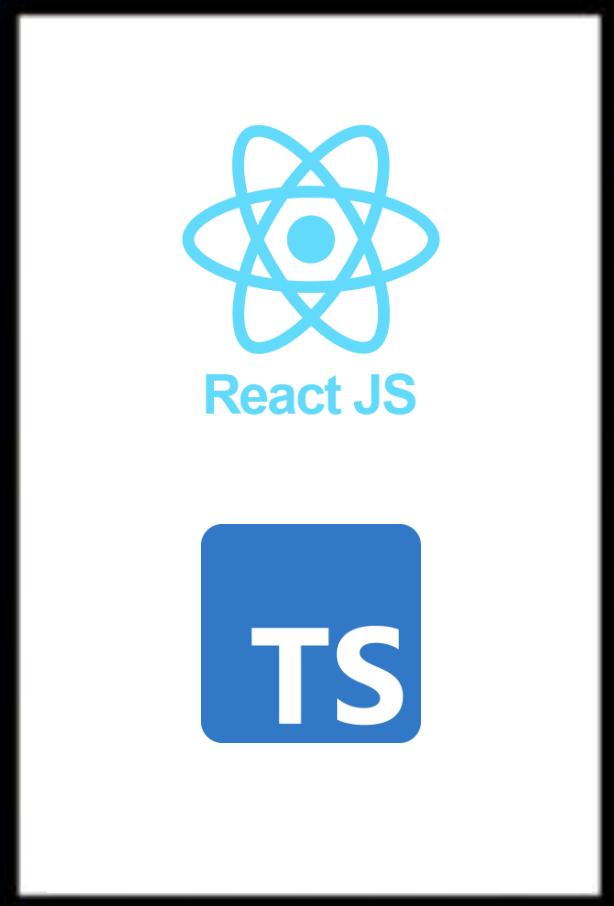
**Section 7:** ใช้ React เชื่อมต่อ Web API พร้อมทำ Authentication



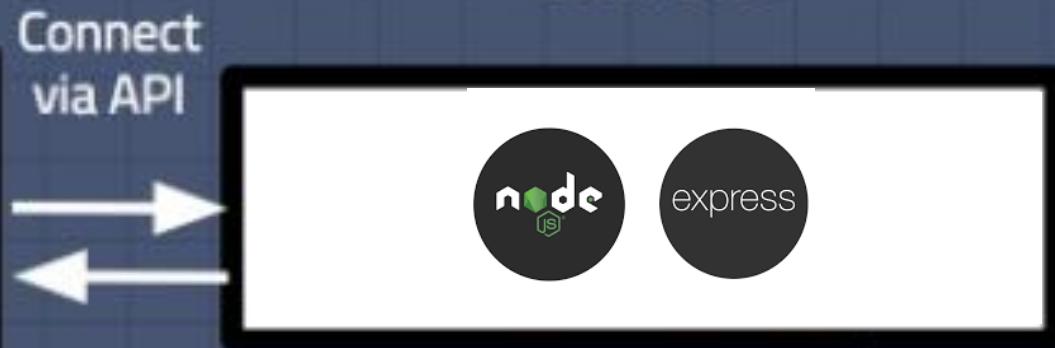
# แนวทางการพัฒนาเว็บแอป



Frontend



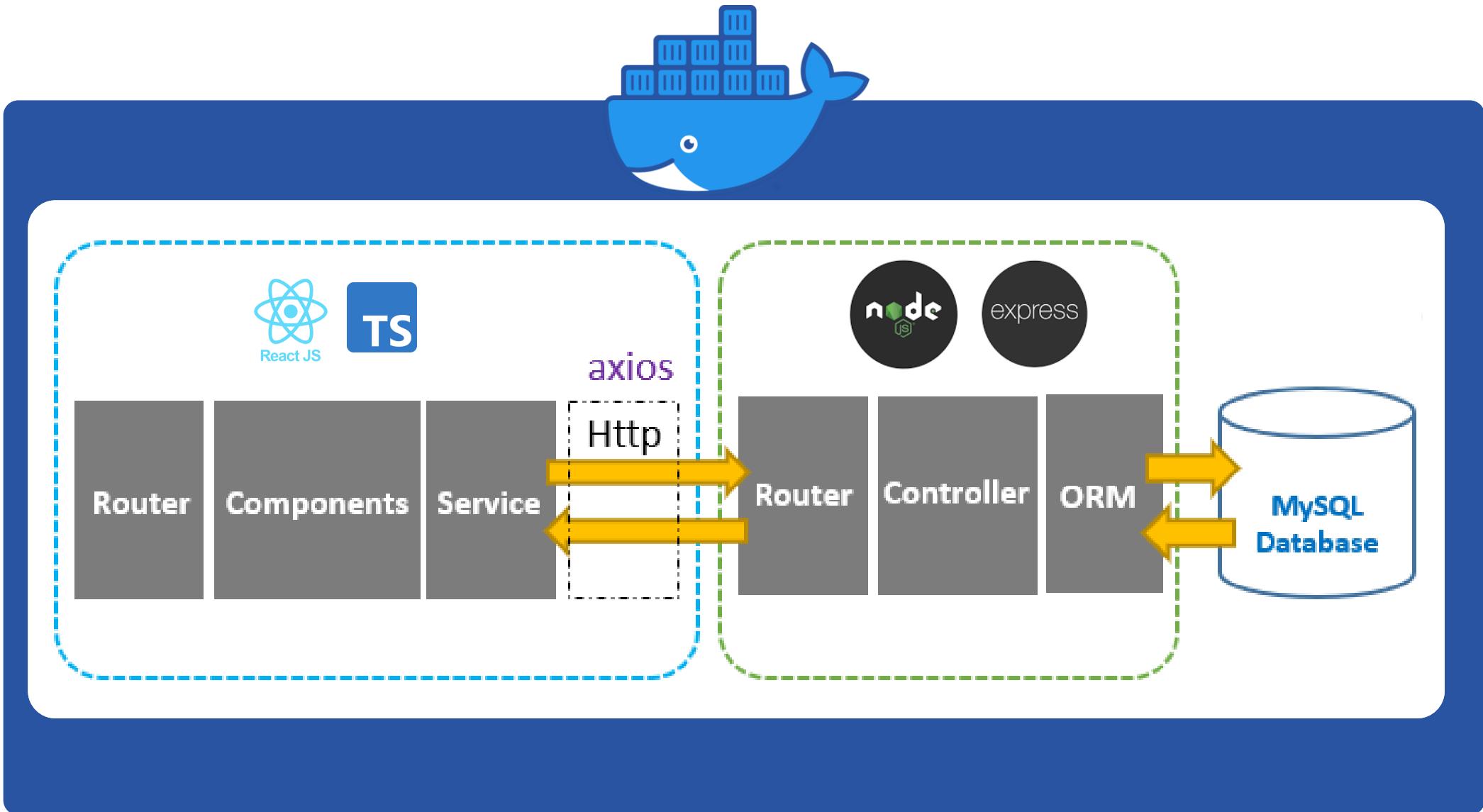
Server

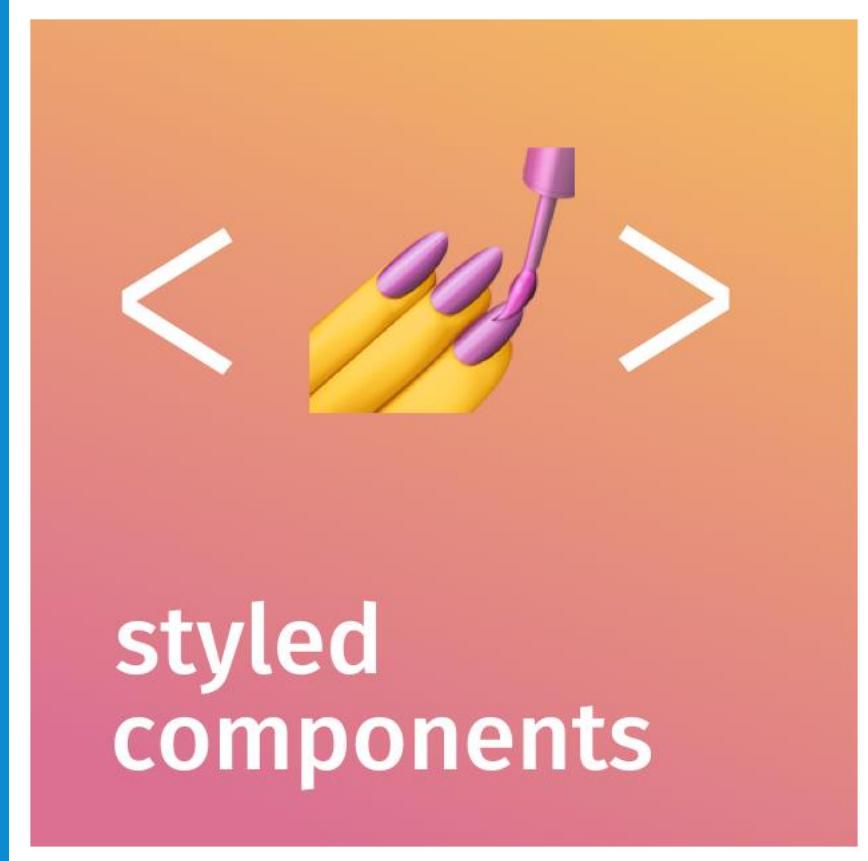


Query & Receive  
the data



# React + Node Stack





## MUI ร่วมกับ Styled-Components ใน React JS

- รู้จัก Styled-Components ใน React js
- ติดตั้ง Styled-Components
- การเรียกใช้งาน ในComponents
- การConfig เพื่อใช้งาน MUI ร่วมกับ Styled-Components  
ตัวอย่างการออกแบบ Template เว็บพอดวัย MUIร่วมกับ  
Styled-Components

# React and NodeJS with Docker Workshop Progress



Day 1



Day 2



Day 3



Day 4



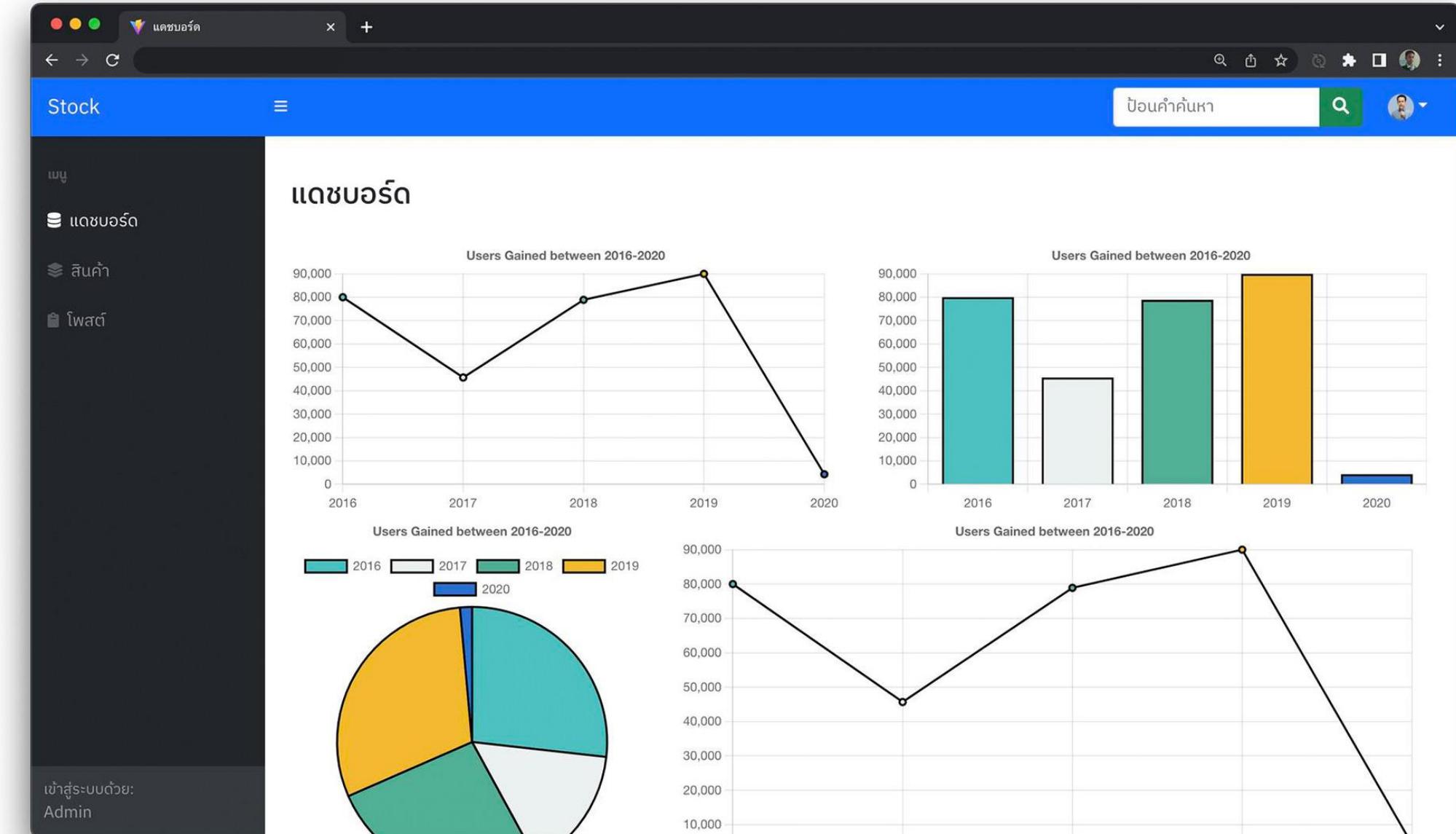
Day 5

# DAY 5

**Section 8:** Workshop React เรียนรู้ CRUD API  
(Create, Read, Update, Delete) ผ่าน API

**Section 9:** การ Build และ Deployed project  
ไปใช้งานจริง





ลิ้นค้า

localhost:5173/product

Stock

เพิ่มสินค้าใหม่

ชื่อสินค้า

Product name

ป้อนชื่อสินค้าก่อน

บาร์โค้ด

Barcode

ป้อนบาร์โค้ดก่อน

รายละเอียด

Price

ป้อนราคา ก่อน

จำนวน

Qty

ป้อนจำนวนชิ้น

รูปภาพ

Image

หมวดหมู่

Electronic

สถานะ

มีตัวอักษร  หมดจากตัวอักษร

เลือกสถานะ ก่อน

ปิด

บันทึก

ป้อนจำนวนชิ้น

+ เพิ่มสินค้าใหม่

บันทึก จัดการ

2023-03-18 Edit Delete

2023-03-18 Edit Delete

2023-03-17 Edit Delete

2023-01-26 Edit Delete

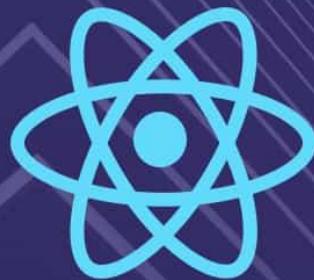
เข้าสู่ระบบด้วย:  
Admin

Copyright © Your Website 2022

Privacy Policy | Terms & Conditions



heroku



## Deploy a React App to Heroku

# Deployment

- การเตรียมความพร้อมของ project ก่อน deploy
- การใช้คำสั่งสำหรับการ build project แบบ production
- แนะนำวิธีการ deploy ไปยัง github page
- แนะนำวิธีการ deploy project ไปยัง Heroku cloud
- แนวทางแก้ไขปัญหาเรื่อง CORS
- แนวทางแก้ไขปัญหาเรื่อง Fallback URL

# Deployment



# Static Server

For environments using [Node](#), the easiest way to handle this would be to install [serve](#) and let it handle the rest:

```
npm install -g serve  
serve -s build
```

The last command shown above will serve your static site on the port **5000**. Like many of [serve](#)'s internal settings, the port can be adjusted using the `-l` or `--listen` flags:

# > npm run build

PROBLEMS 5 OUTPUT DEBUG CONSOLE TERMINAL

1: cmd

```
F:\ReactJSOnline\react-webapi\build>npm run build
> react-webapi@0.1.0 build F:\ReactJSOnline\react-webapi
> react-scripts build

Creating an optimized production build...
Compiled successfully.

File sizes after gzip:

 50.53 KB  build\static\js\2.aaf61152.chunk.js
 38.66 KB  build\static\css\main.a1fee737.chunk.css
 4.07 KB   build\static\js\main.b63290d7.chunk.js
 776 B     build\static\js\runtime-main.170107c9.js

The project was built assuming it is hosted at /.
You can control this with the homepage field in your package.json.

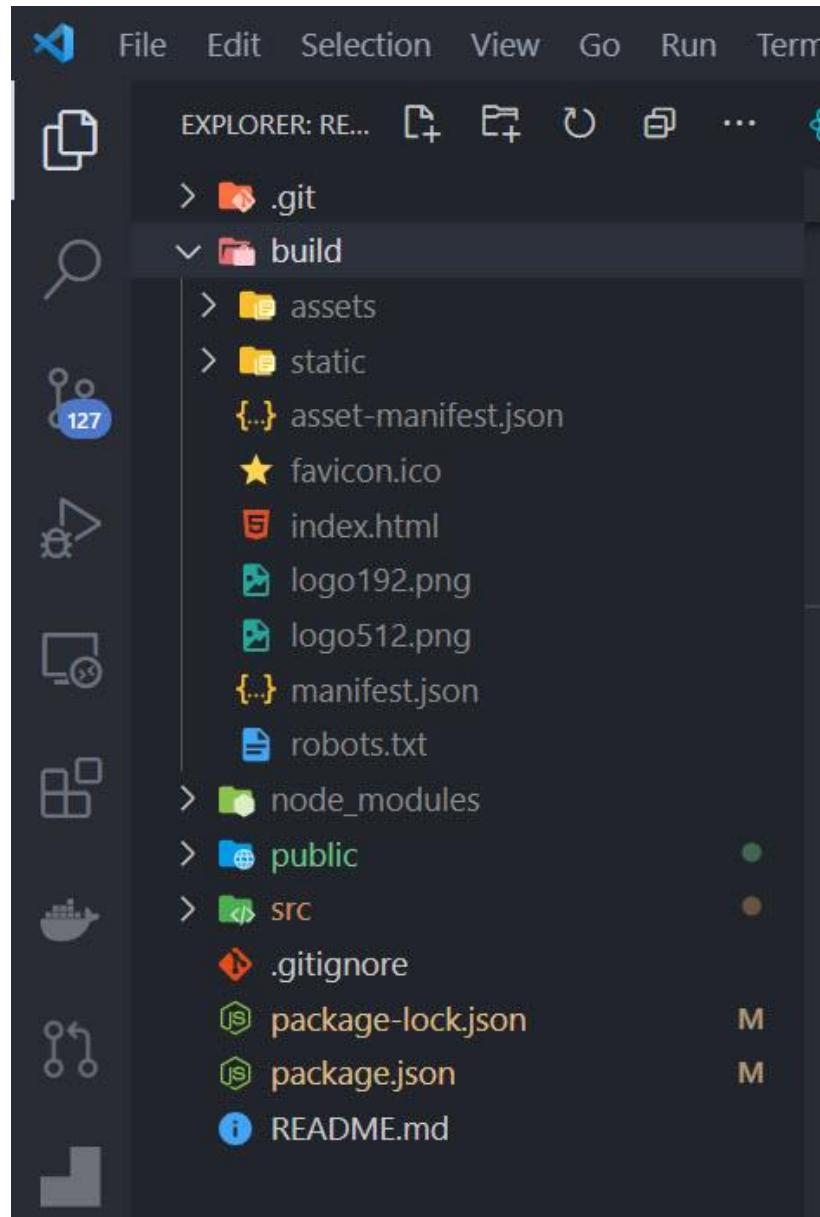
The build folder is ready to be deployed.
You may serve it with a static server:

  serve -s build

Find out more about deployment here:

  https://cra.link/deployment

F:\ReactJSOnline\react-webapi\build>
```



# > serve -s build

The image shows a web browser window with two main panes. On the left, a registration form titled "ลงทะเบียน" is displayed. It has four input fields: "ชื่อ-สกุล", "อีเมล์", "ชื่อผู้ใช้", and "รหัสผ่าน", followed by two buttons: "ลงทะเบียน" and "เข้าสู่ระบบ". On the right, a Lighthouse performance audit report for the URL <http://localhost:5000/register> is shown. The report includes scores for Performance (99), Accessibility (100), Best Practices (100), SEO (100), and PWA (Progressive Web App). A note at the top states: "There were issues affecting this run of Lighthouse: • There may be stored data affecting loading performance in this location: IndexedDB. Audit this page in an incognito window to prevent those resources from affecting your scores." Below this, a large "Performance" score of 99 is highlighted, with detailed metrics listed below it.

ลงทะเบียน

ชื่อ-สกุล

อีเมล์

ชื่อผู้ใช้

รหัสผ่าน

ลงทะเบียน

เข้าสู่ระบบ

Performance

Accessibility

Best Practices

SEO

PWA

There were issues affecting this run of Lighthouse:

- There may be stored data affecting loading performance in this location: IndexedDB. Audit this page in an incognito window to prevent those resources from affecting your scores.

Metrics

First Contentful Paint	0.5 s	Time to Interactive	0.5 s
Speed Index	0.5 s	Total Blocking Time	0 ms

# Deployed to Apache



The screenshot shows a code editor with multiple tabs at the top, including Register.jsx, package.json, index.js, Forgotpassword.jsx, app.css, Counter.jsx, routes.js, and others. The package.json tab is active, displaying the following JSON code:

```
    "dependencies": {  
        "@testing-library/react": "^11.2.5",  
        "@testing-library/user-event": "^12.7.1",  
        "react": "^17.0.1",  
        "react-document-title": "^2.0.3",  
        "react-dom": "^17.0.1",  
        "react-router-dom": "^5.2.0",  
        "react-scripts": "4.0.2",  
        "sweetalert2": "^10.15.5",  
        "web-vitals": "^1.1.0"  
    },  
    "homepage": "http://localhost/reactwebapi/",  
    "scripts": {  
        "start": "react-scripts start",  
        "build": "react-scripts build",  
        "test": "react-scripts test",  
        "eject": "react-scripts eject"  
    },  
    "eslintConfig": {  
        "extends": [  
            "react-app",  
            "react-app/jest"  
        ]  
    },  
    "browserslist": {  
        "production": [  
        ]  
    }  
}
```

The screenshot shows a code editor interface with multiple tabs at the top, each representing a file: Register.jsx, package.json, index.js (selected), Forgotpassword.jsx, app.css, Counter.jsx, routes.jsx, and three others with icons. The main area displays the content of the index.js file.

```
src > index.js
  3 import {BrowserRouter as Router} from 'react-router-dom'
  4 import routes from './routes'
  5 import './assets/css/app.css'
  6 import './styles/custom.css'
  7
  8 ReactDOM.render(
  9   <React.StrictMode>
 10    <Router basename={'/reactwebapi'}>
 11      { routes }
 12    </Router>
 13  </React.StrictMode>,
 14  document.getElementById('root')
 15)
 16
```

```
F:\ReactJSOnline\react-webapi>npm run build
```

```
> react-webapi@0.1.0 build F:\ReactJSOnline\react-webapi
> react-scripts build
```

```
Creating an optimized production build...
Compiled successfully.
```

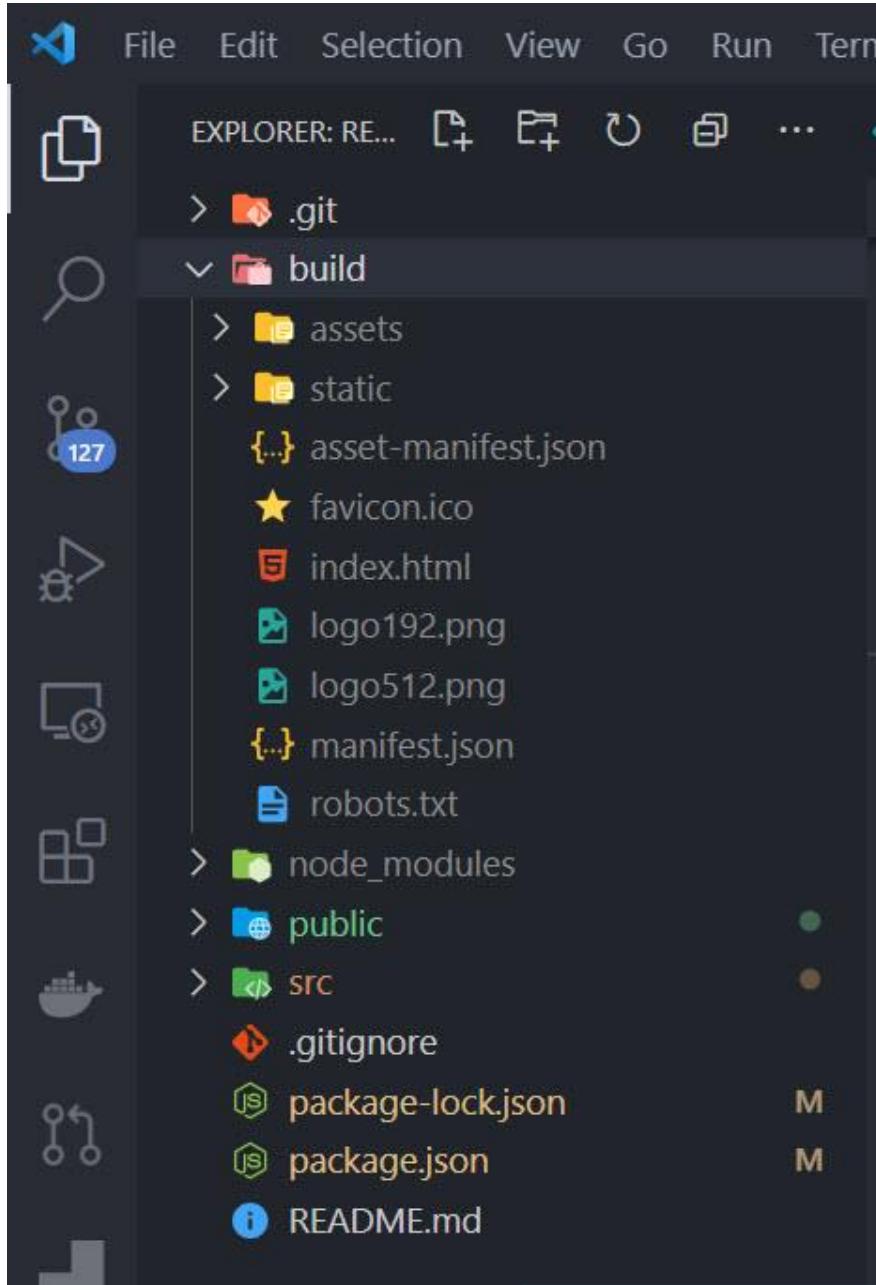
```
File sizes after gzip:
```

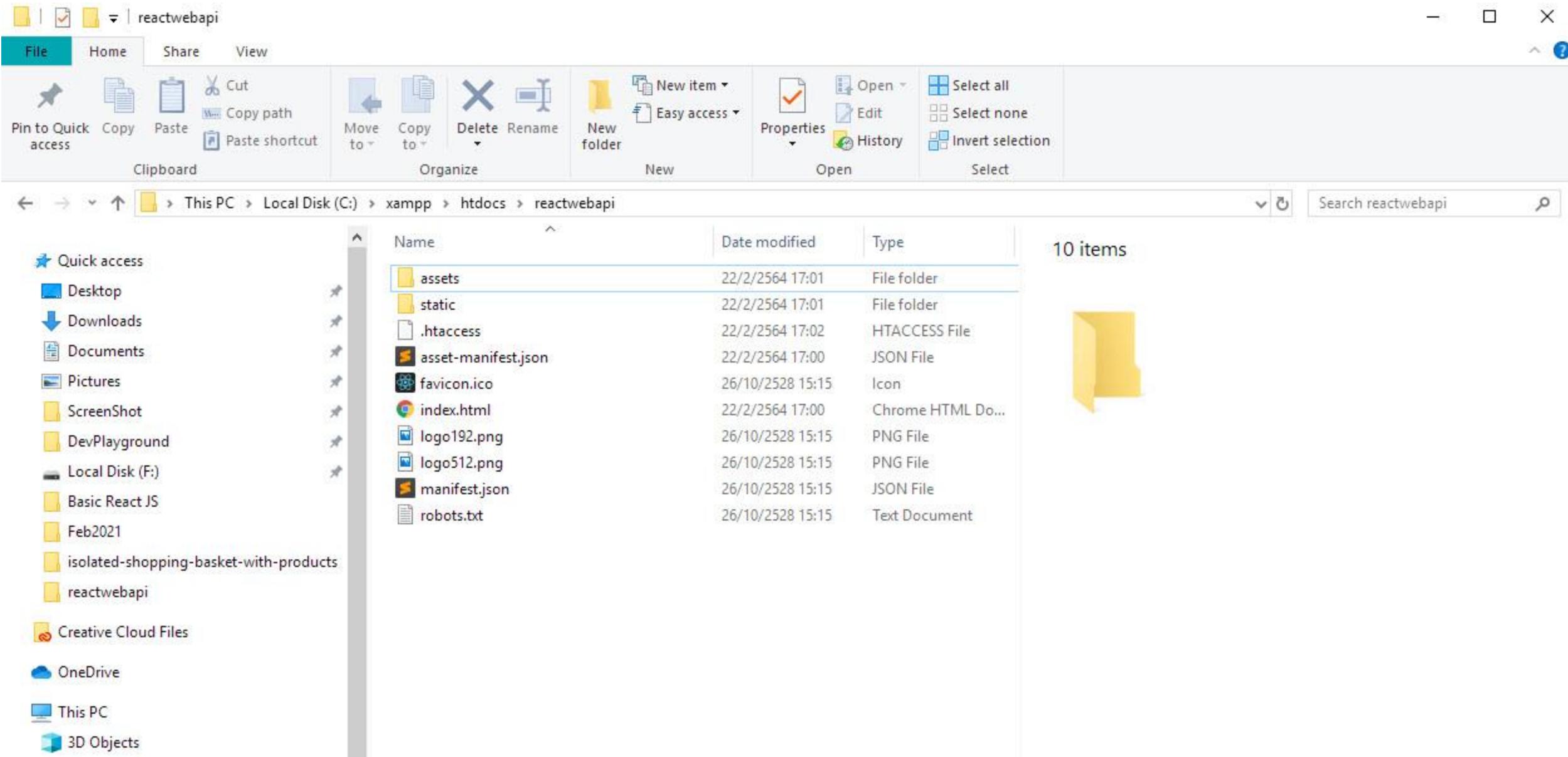
```
50.53 KB  build\static\js\2.aaf61152.chunk.js
38.69 KB  build\static\css\main.3a806a39.chunk.css
4.08 KB   build\static\js\main.1f727981.chunk.js
785 B     build\static\js\runtime-main.3e557204.js
```

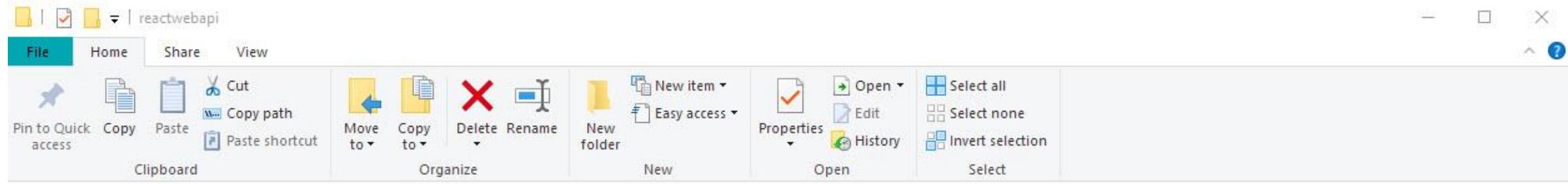
```
The project was built assuming it is hosted at /reactwebapi/.
```

ning..

Ln 16, Col 1 Spaces: 2 UTF-8 LF JavaScript React ✓ ESLint ✓ Prettier ⌂







← → ⌂ ⌃ ⌄ This PC > Local Disk (C:) > xampp > htdocs > reactwebapi

Name	Date modified	Type	.htaccess
assets	22/2/2564 17:01	File folder	
static	22/2/2564 17:01	File folder	
.htaccess	22/2/2564 17:02	HTACCESS File	 Date modified: 22/2/2564 17:02 Size: 108 bytes Date created: 22/2/2564 16:57
asset-manifest.json	22/2/2564 17:00	JSON File	
favicon.ico	26/10/2528 15:15	Icon	
index.html	22/2/2564 17:00	Chrome HTML Do...	
logo192.png	26/10/2528 15:15	PNG File	
logo512.png	26/10/2528 15:15	PNG File	
manifest.json	26/10/2528 15:15	JSON File	
robots.txt	26/10/2528 15:15	Text Document	

EditPlus [Default] - [C:\xampp\htdocs\reactwebapi\.htaccess]

File Edit View Document Project Tools Browser Window Help

Directory Cliptext

[C:]

C:\xampp\htdocs\reactwebapi

```
1 Options -MultiViews
2 RewriteEngine On
3 RewriteCond %{REQUEST_FILENAME} !-
4 RewriteRule ^ index.html [QSA,L]
```

## ลงทะเบียน

ชื่อ-สกุล

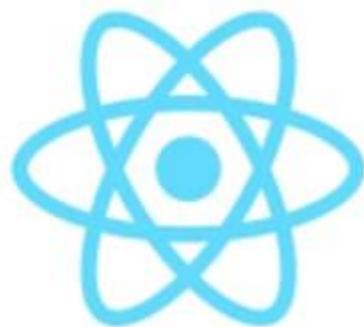
อีเมล

ชื่อผู้ใช้

รหัสผ่าน

ลงทะเบียน[เข้าสู่ระบบ](#)

# Deployed to Github Page



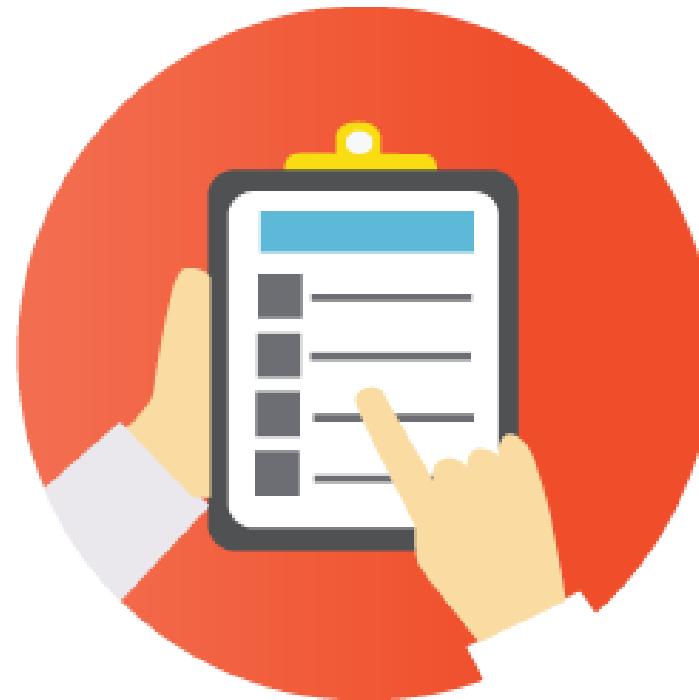
+



**Create React App**

**Github Pages**

# แบบทดสอบหลังอบรม

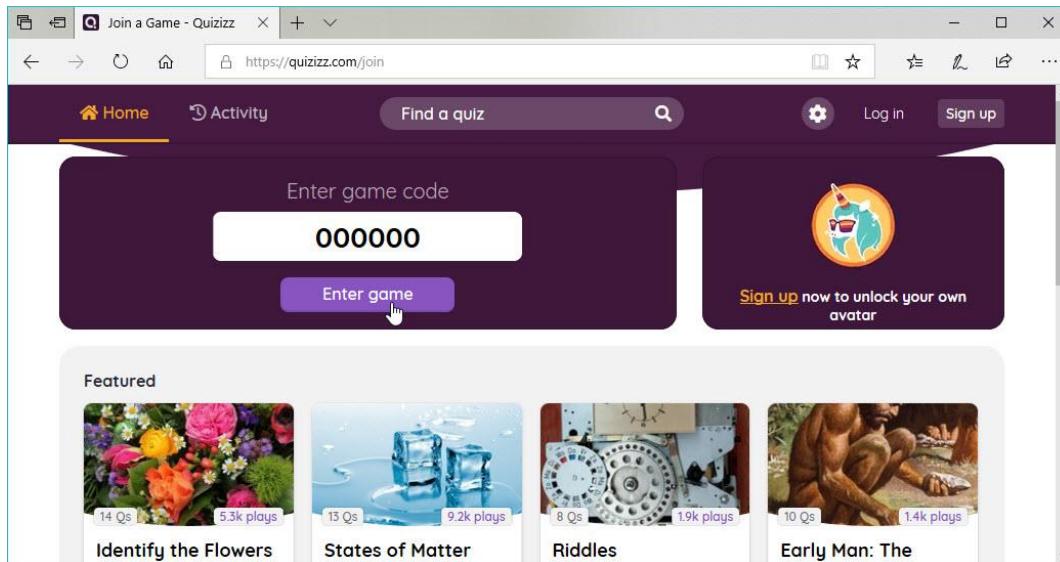


Pretest

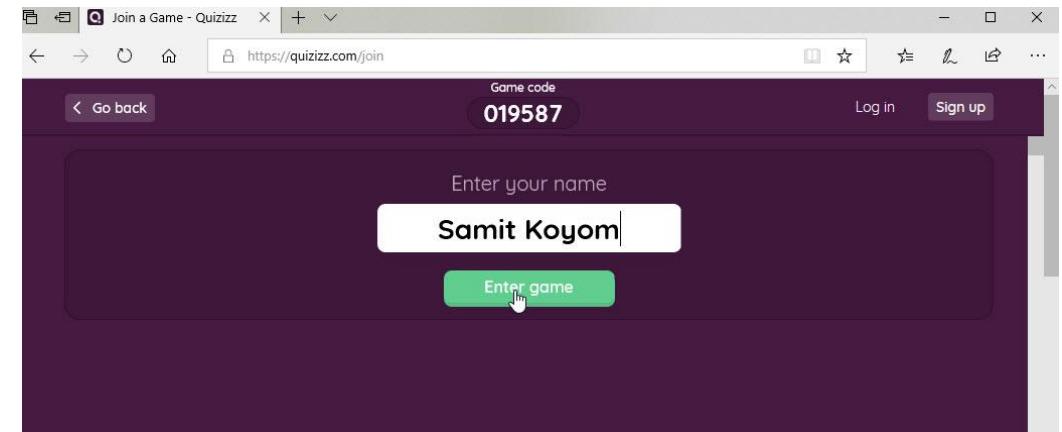
# Posttest ทำแบบทดสอบหลังเรียน

STEP 1: เข้าทำแบบทดสอบที่ลิงก์ ป้อนรหัสเข้าห้องสอบ

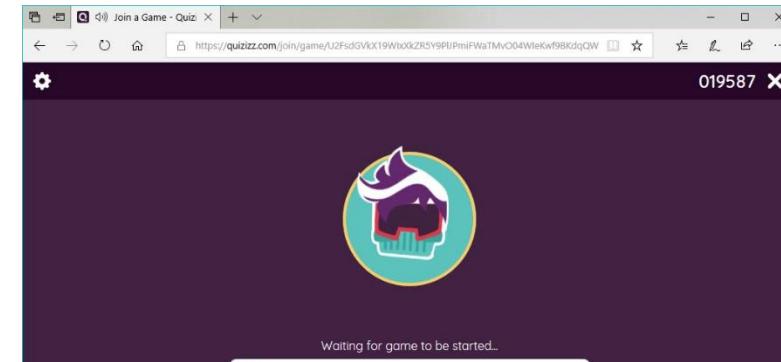
[quizizz.com/join](https://quizizz.com/join)



STEP 2: ป้อนชื่อ



STEP 3: รอผู้สอน Start ข้อสอบ





# อาจารย์สามิต โกยม



samitkoyom@gmail.com



iamsamit



facebook.com/iamsamit



twitter.com/iamsamit



github.com/iamsamitdev



linkedin.com/in/samit-koyom-72173348