

# Web App Development

DAY 9 - ROAD TO FRONT-END DEVELOPER BOOTCAMP

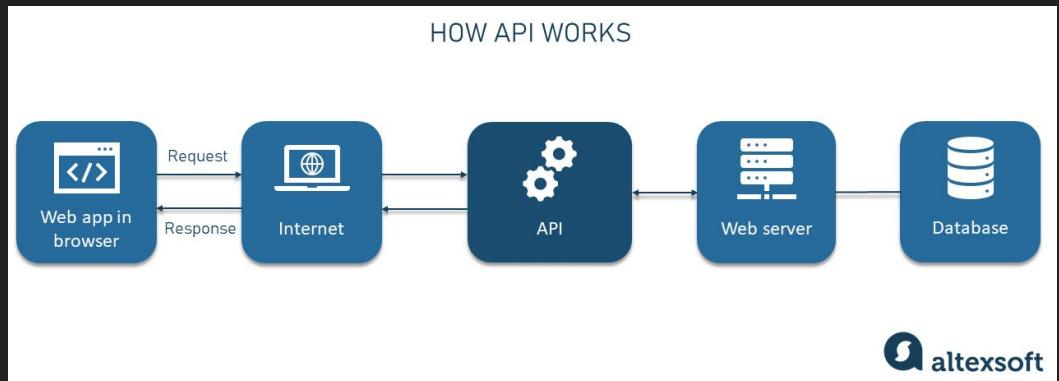
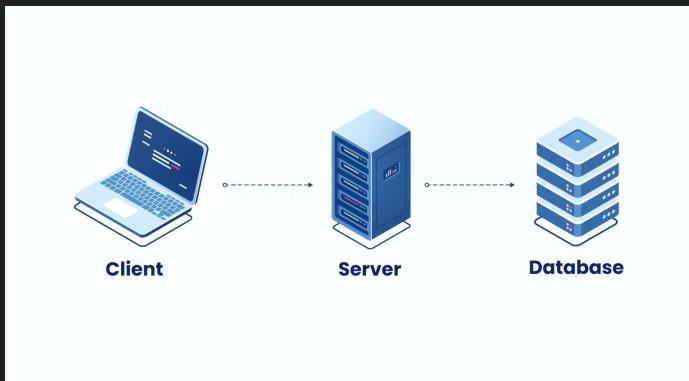
เราเรียนอะไรกันไปบ้าง ?

# HTML CSS JavaScript

FE : HTML CSS JavaScript

แล้ว Back-end หละ เขาทำอะไร ?

# แล้ว Back-end หละ เข้าทำอะไร ?



# ນາງົງຈັກກັບ Express

The screenshot shows the official Express.js website. At the top, there's a black banner with the text "Black Lives Matter." and "Support the Equal Justice Initiative." Below the banner, the Express logo is on the left, followed by a search bar and a navigation menu with links to Home, Getting started, Guide, API reference, Advanced topics, and Resources. The main title "Express 4.18.1" is prominently displayed, followed by the subtitle "Fast, unopinionated, minimalist web framework for Node.js". A command line interface (CLI) command "\$ npm install express --save" is shown in a box. A yellow callout box contains the text "Express 5.0 beta documentation is now available. The beta API documentation is a work in progress. For information on what's in the release, see the Express release history." Below this, there are four sections: "Web Applications", "APIs", "Performance", and "Frameworks". Each section has a brief description and a link to more information. At the bottom, there's footer text about documentation translations and a Creative Commons license notice.

Black Lives Matter.  
Support the Equal Justice Initiative.

Express

search

Home Getting started Guide API reference Advanced topics Resources

Express 4.18.1

Fast, unopinionated, minimalist web framework for Node.js

\$ npm install express --save

Express 5.0 beta documentation is now available. The beta API documentation is a work in progress. For information on what's in the release, see the Express release history.

Web Applications

Express is a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications.

APIs

With a myriad of HTTP utility methods and middleware at your disposal, creating a robust API is quick and easy.

Performance

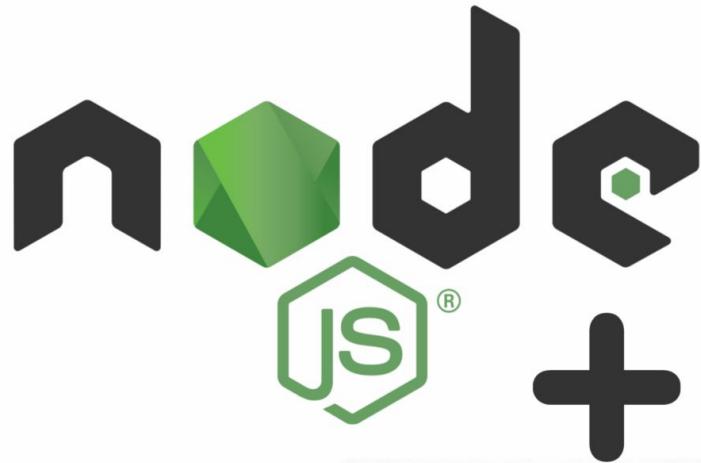
Express provides a thin layer of fundamental web application features, without obscuring Node.js features that you know and love.

Frameworks

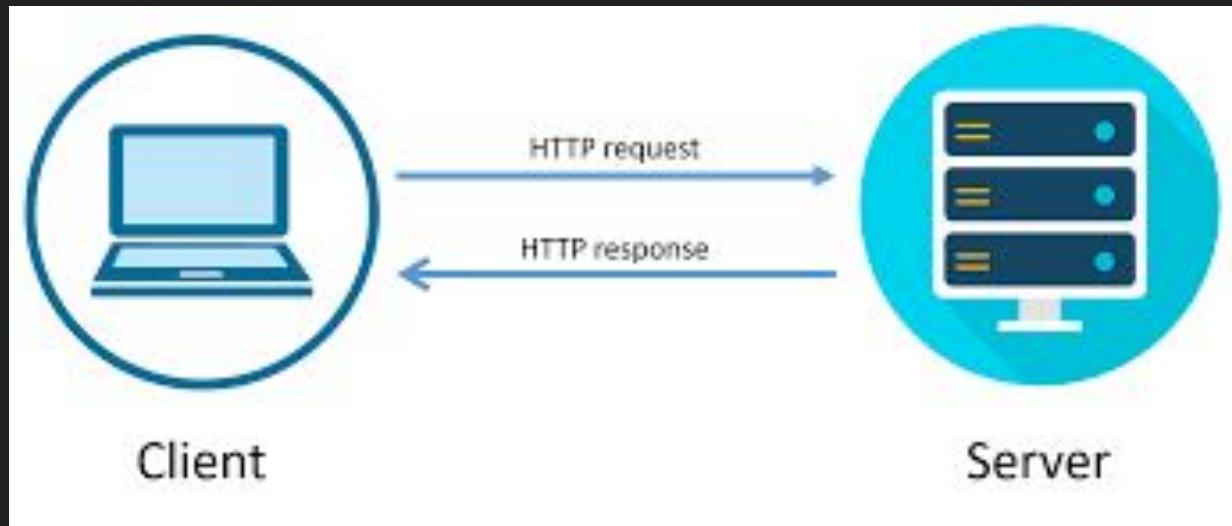
Many popular frameworks are based on Express.

Documentation translations provided by StrongLoop/IBM: French, German, Spanish, Italian, Japanese, Russian, Chinese, Traditional Chinese, Korean, Portuguese. Community translation available for: Slovak, Ukrainian, Uzbek, Turkish and Thai.

Star Express is a project of the OpenS Foundation. Fork the website on GitHub. Copyright © 2017 StrongLoop, IBM, and other expressjs.com contributors. This work is licensed under a Creative Commons Attribution-ShareAlike 3.0 United States License.



Express



# ນາງົງຈັກກັບ Express

The screenshot shows the official Express.js website. At the top, there's a black banner with the text "Black Lives Matter." and "Support the Equal Justice Initiative." Below the banner, the Express logo is on the left, followed by a search bar and a navigation menu with links to Home, Getting started, Guide, API reference, Advanced topics, and Resources.

The main content area features the title "Express 4.18.1" and the subtitle "Fast, unopinionated, minimalist web framework for Node.js". Below this, there's a command-line interface (CLI) example: "\$ npm install express --save". A yellow callout box contains the message "Express 5.0 beta documentation is now available. The beta API documentation is a work in progress. For information on what's in the release, see the Express release history.".

The page is divided into four main sections: "Web Applications", "APIs", "Performance", and "Frameworks". Each section has a brief description and a link to more information. At the bottom, there's footer text about documentation translations and a Creative Commons license notice.

Documentation translations provided by StrongLoop/IBM: French, German, Spanish, Italian, Japanese, Russian, Chinese, Traditional Chinese, Korean, Portuguese.  
Community translation available for: Slovak, Ukrainian, Uzbek, Turkish and Thai.

Star Fork This work is licensed under a Creative Commons Attribution-ShareAlike 3.0 United States License.

# ຝຶເຈອຣ໌ຫລັກ

## 1. Middleware

Express allows you to use middleware functions to perform actions on the request and response objects, or to end the request-response cycle. Middleware functions can also call the next middleware in the stack.

## 2. Routing

Express provides a routing table for handling different HTTP methods (GET, POST, PUT, DELETE, etc.) and paths (URLs). This makes it easy to define how your application should respond to different types of client requests.

## 3. Templating

Express supports various templating engines like EJS, Pug, and Handlebars, allowing you to generate dynamic HTML content on the server-side.

## 4. Error Handling

Express has built-in error-handling mechanisms that make it easier to catch and handle errors gracefully.

## 5. Static Files

You can serve static files like images, CSS, and JavaScript directly from designated folders, typically named public.

## 6. Modular

Express is very modular and extensible. You can easily plug in different middleware or libraries to add more functionalities like parsing cookies, handling file uploads, or implementing authentication.

# Why Use Express.js?

Simplicity: It's easy to set up and get a server running.

Flexibility: It doesn't enforce a specific way of doing things, allowing you to structure your app as you see fit.

Community Support: Being one of the most popular Node.js frameworks means a lot of community-contributed middleware and tutorials are available.

Performance: Built on Node.js, it inherits its performance capabilities, making it suitable for I/O-bound and data-intensive real-time applications.

Scalability: It's easy to scale horizontally by running multiple instances behind a load balancer.

# Why Use Express.js?

Simplicity: It's easy to set up and get a server running.

Flexibility: It doesn't enforce a specific way of doing things, allowing you to structure your app as you see fit.

Community Support: Being one of the most popular Node.js frameworks means a lot of community-contributed middleware and tutorials are available.

Performance: Built on Node.js, it inherits its performance capabilities, making it suitable for I/O-bound and data-intensive real-time applications.

Scalability: It's easy to scale horizontally by running multiple instances behind a load balancer.

# Example Use Cases

APIs: Build RESTful APIs for various client-side applications.

Single Page Applications: Serve as the backend for React, Angular, or Vue.js apps.

Streaming Services: Handle real-time data streaming.

Authentication Services: Implement OAuth or JWT-based authentication.

Microservices: Build small, focused microservices for larger applications.

a coding AI directly in your IDE.

## Create a Repl

Import from GitHub



Template

express



Express.js  
replit

Express.js + htmx  
replit

Express.js TODOS Backend Server  
replit

Hyper Express  
7heMech

Go  
replit

PyScript  
HamasAkhtar

Blockly Typescript

Title

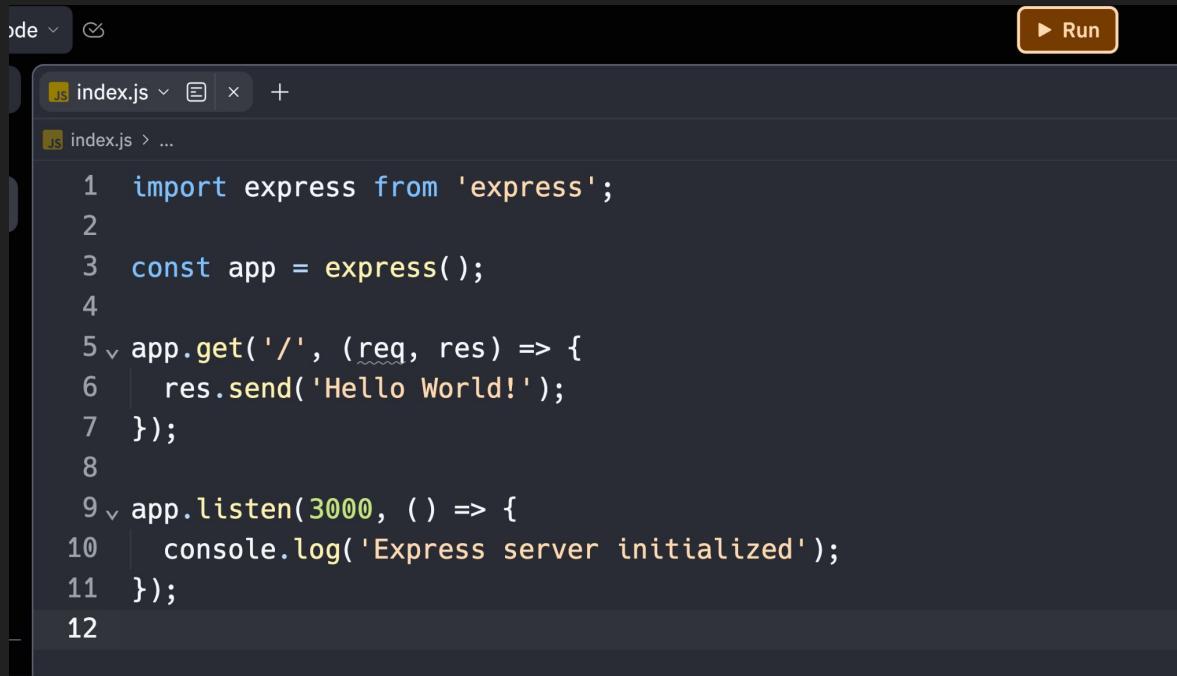
Name your Repl

Public

Anyone can view and fork this Repl.

Upgrade to make private

+ Create Repl



```
1 import express from 'express';
2
3 const app = express();
4
5 app.get('/', (req, res) => {
6   res.send('Hello World!');
7 });
8
9 app.listen(3000, () => {
10   console.log('Express server initialized');
11 });
12
```

เมื่อเรียบร้อยให้กดปุ่ม Run

The screenshot shows a browser window with two tabs. The left tab is titled 'index.js' and contains the following code:

```
1 import express from 'express';
2
3 const app = express();
4
5 app.get('/', (req, res) => {
6   res.send('Hello World!');
7 });
8
9 app.listen(3000, () => {
10   console.log('Express server initialized');
11});
```

The right tab is titled 'Webview' and displays the URL <https://unripefortunatecharactercode.borntodev.repl.co>. The page content is "Hello World!". The top right corner of the browser window shows the 'BORTO DEV' logo.

เราจะพบร้านเว็บของเรารากฐานแล้ว

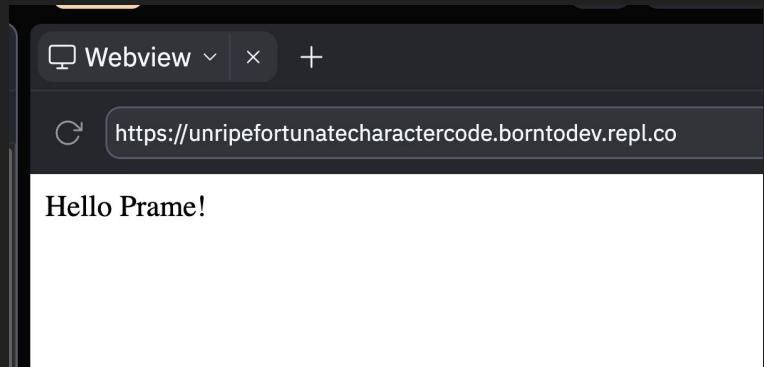
```
@app.route('/')
def index():
    return 'Hello from Flask!'
```

ทดลองเปลี่ยนข้อความตรงนี้ดูกัน !

```
5 app.get('/', (req, res) => {  
6   res.send('Hello World!');  
7 });
```

ทดลองเปลี่ยนข้อความตรงนี้ดูกัน !

```
5 app.get('/', (req, res) => {  
6   res.send('Hello Prame!');  
7 });  
8
```

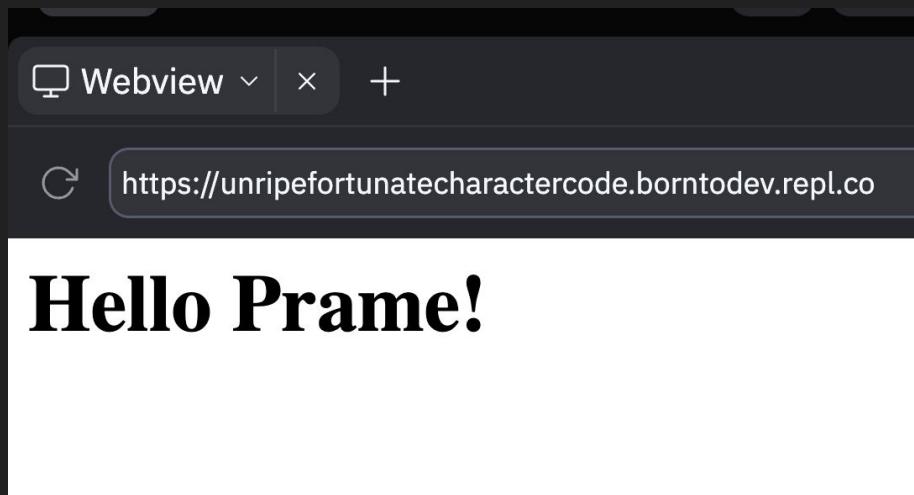


สิ่งเดตดูว่าข้อความจะถูกเปลี่ยนไปแล้ว

```
5 v app.get('/', (req, res) => {  
6   res.send('<h1>Hello Prame!  
7   </h1>');  
8});
```

ทดสอบใส่ Tag HTML ดูกัน !

```
5 v app.get('/', (req, res) => {  
6   res.send('<h1>Hello Prame!  
7   </h1>');  
7});
```



ເຮືອບຮ້ອຍຍ !

```
5 v app.get( '/', (req, res) => {  
6   res.send( '<h1>Hello Prame!</h1>' );  
7 });
```

ว่าแต่ app.get อะไรมีคืออะไรกัน ?

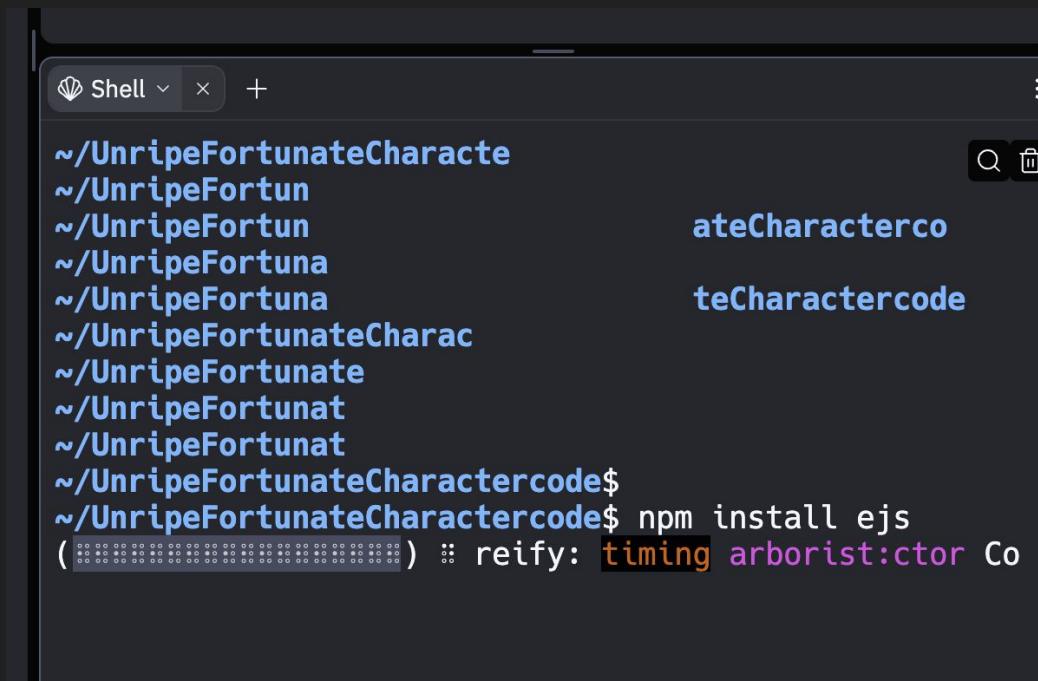
งั้นแปลว่าเราเอาทั้งเว็บใส่ไปในนี้  
ก็จะได้เลยใช่ไหม ?

```
5 app.get( '/', (req, res) => {  
6   res.send( '<h1>Hello Prame!</h1>' );  
7 });
```

จั้นแปลว่าเราเอาทั้งเว็บไซต์ไปในนี้  
ก็จะได้เลยใช่ไหม ?

คำตอบคือ ใช่ ! แต่มันมีวิธีที่ดีกว่านี้耶อะอยู่ !

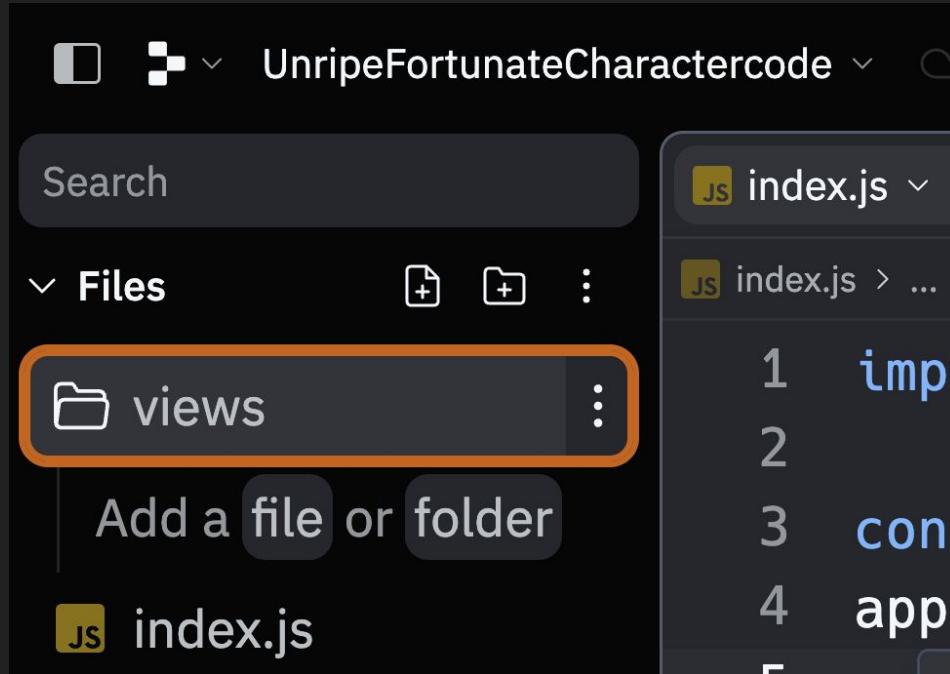
เรารอเรียกสิ่งนี้ว่าการใช้งาน  
Template Engine



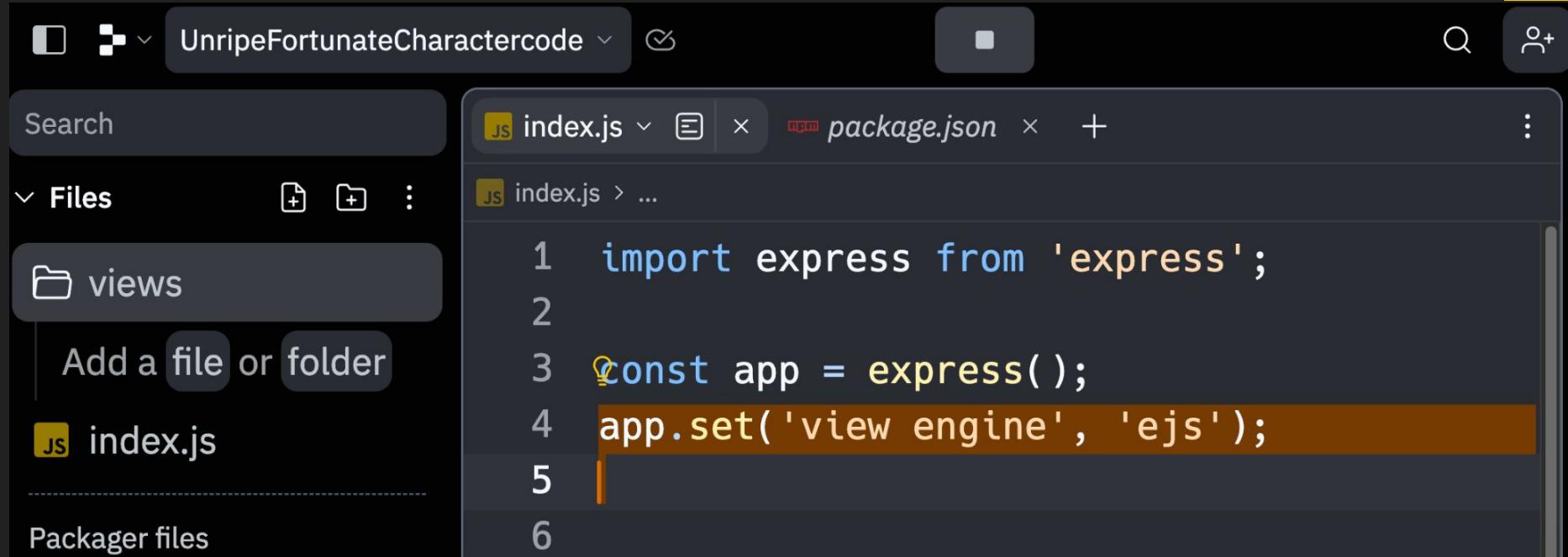
```
Shell ~/UnripeFortunateCharactercode$ ~/UnripeFortunateCharactercode$ npm install ejs  
(:): reify: timing arborist:ctor Co
```

ติดตั้ง ejs ผ่าน shell ด้วยคำสั่ง

npm install ejs



สร้างโฟลเดอร์ views ขึ้นมา



The screenshot shows a code editor interface with a dark theme. At the top, there's a header bar with icons for file operations and a search bar. Below the header is a sidebar labeled "Files" which contains a folder named "views", a button to "Add a file or folder", and a file named "index.js". The main area of the editor displays two tabs: "index.js" and "package.json". The "index.js" tab is active and shows the following code:

```
1 import express from 'express';
2
3 const app = express();
4 app.set('view engine', 'ejs');
5
6
```

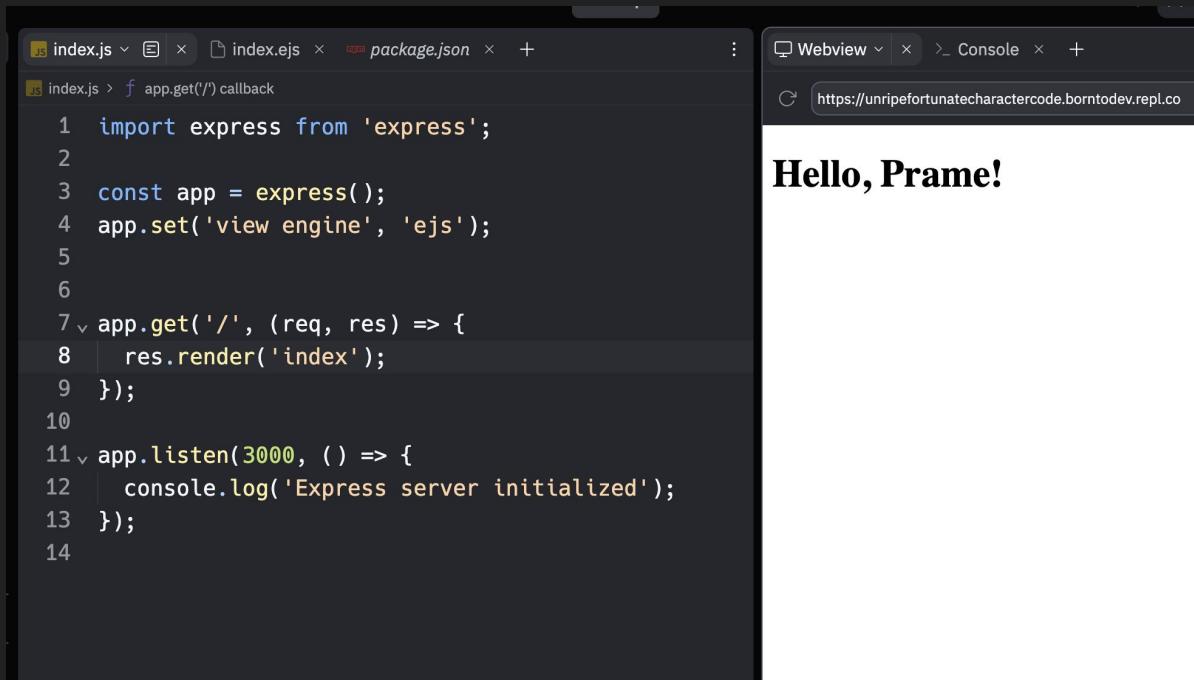
The code uses ES6 syntax, including imports and arrow functions. The "ejs" string in the fourth line is highlighted with a yellow background. The "views" folder in the sidebar is also highlighted with a yellow background.

ເຮີຍກິ່ຈີ view engine ກັບ ejs

The screenshot shows a dark-themed code editor interface. On the left, a sidebar lists files: 'views' (containing 'index.ejs'), 'index.js', and 'package.json'. The 'index.ejs' file is selected and highlighted. The main area is a code editor showing the contents of 'index.ejs':

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>Welcome</title>
5 </head>
6 <body>
7   <h1>Hello, Prame!</h1>
8 </body>
9 </html>
```

สร้างไฟล์ index.ejs ไว้ใน views



```
index.js index.ejs package.json +  
index.js > f app.get('/') callback  
1 import express from 'express';  
2  
3 const app = express();  
4 app.set('view engine', 'ejs');  
5  
6  
7 app.get('/', (req, res) => {  
8   res.render('index');  
9 });  
10  
11 app.listen(3000, () => {  
12   console.log('Express server initialized');  
13 });  
14
```

Hello, Prame!

ปรับแก้ app.get('/') เป็น render

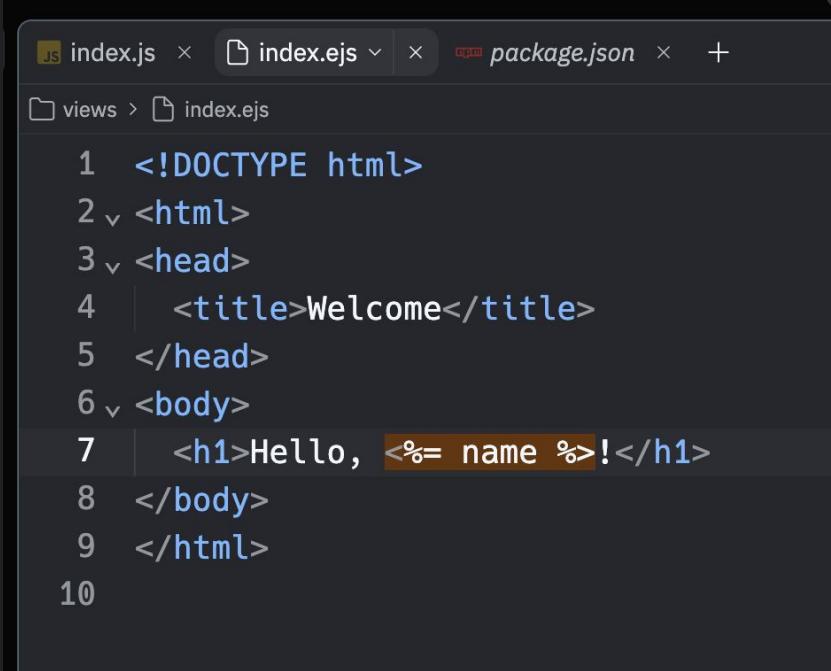


```
index.js index.ejs package.json +  
index.js > f app.get('/').callback  
1 import express from 'express';  
2  
3 const app = express();  
4 app.set('view engine', 'ejs');  
5  
6  
7 app.get('/', (req, res) => {  
8   res.render('index');  
9 });  
10  
11 app.listen(3000, () => {  
12   console.log('Express server initialized');  
13 });  
14
```

Hello, Prame!

ເວັບກີໄສສົດງແລ້ວ

Template Engine มีเอาไว้ทำ  
Dynamic Data ลงในไฟล์ HTML ของเรา



```
index.js × index.ejs × package.json × +  
views > index.ejs  
1  <!DOCTYPE html>  
2  <html>  
3  <head>  
4  |   <title>Welcome</title>  
5  </head>  
6  <body>  
7  |   <h1>Hello, <%= name %>!</h1>  
8  </body>  
9  </html>  
10
```

ลองเพิ่ม โค้ดนี้ ลงไป



The screenshot shows a code editor window with a dark theme. The tab bar at the top has four tabs: 'index.js' (selected), 'index.ejs', 'package.json', and a '+' icon. Below the tabs, a sidebar shows 'index.js > ...'. The main code area contains the following JavaScript code:

```
1 import express from 'express';
2
3 const app = express();
4 app.set('view engine', 'ejs');
5
6
7 app.get('/', (req, res) => {
8   res.render('index', { name: 'Prame' });
9 })
```

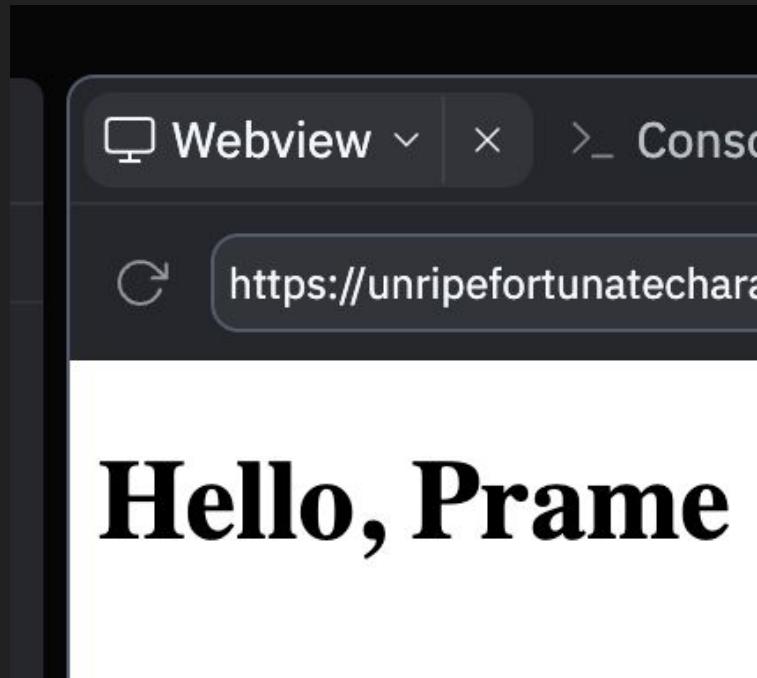
ลองเพิ่ม โค้ดนี้ ลงไป



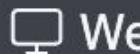
The screenshot shows a code editor window with a dark theme. The tab bar at the top has four tabs: 'index.js' (selected), 'index.ejs', 'package.json', and a '+' icon. Below the tabs, a sidebar shows 'index.js > ...'. The main code area contains the following JavaScript code:

```
1 import express from 'express';
2
3 const app = express();
4 app.set('view engine', 'ejs');
5
6
7 app.get('/', (req, res) => {
8   res.render('index', { name: 'Prame' });
9 })
```

ลองเพิ่ม โค้ดนี้ ลงไป



ลองเพิ่ม โค้ดนี้ ลงไป



Webview



&gt;\_ Console

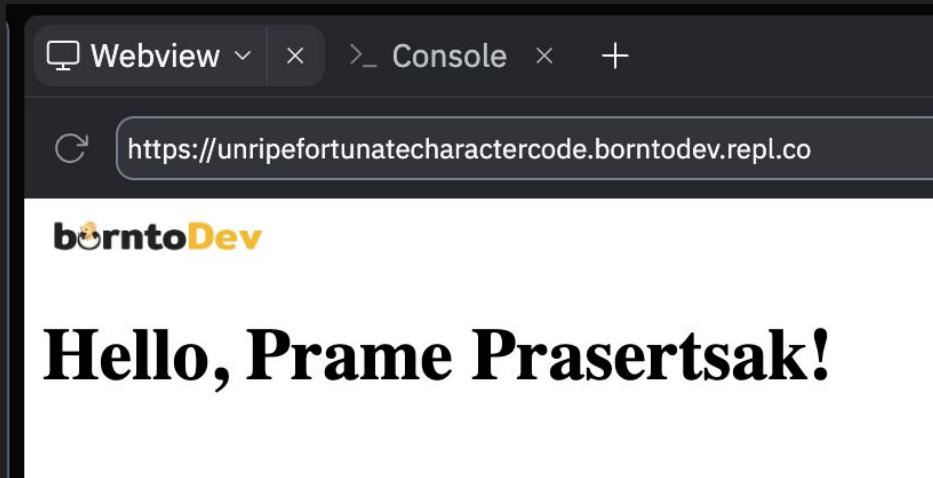
<https://unripefortunatecharactercode.borntodev.repl.co>

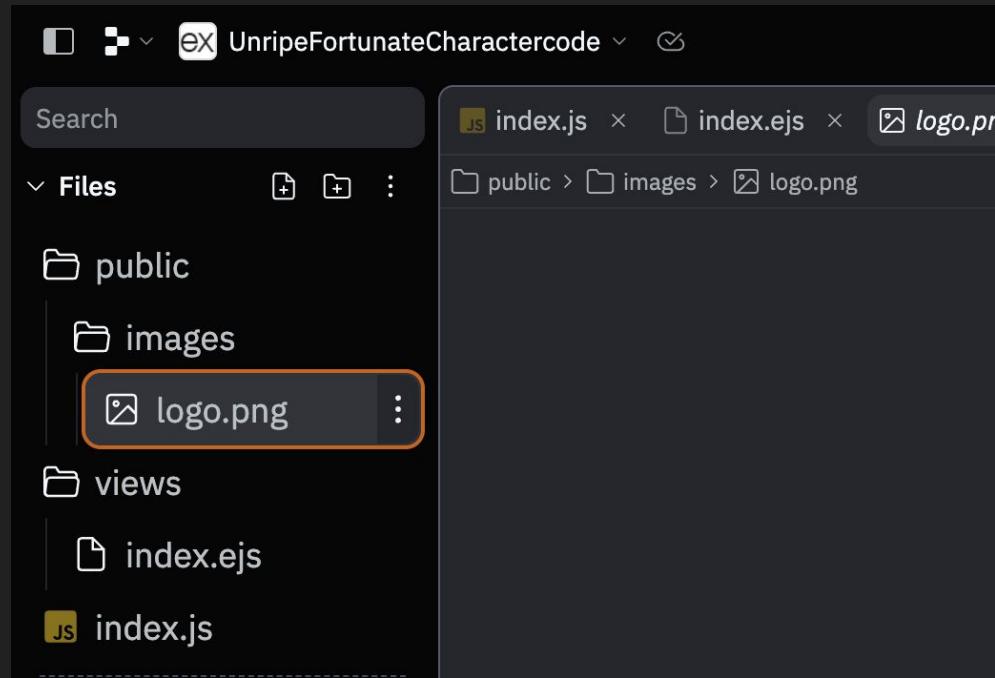
# Hello, Prame Prasertsak!

หากเราต้องการส่งข้อมูลภายใน Object

ไป 2 ตัว และแสดง ชื่อ + นามสกุลหละ ?

# การนำ Static Content มาแสดง ?





สร้าง Folder : public > images  
และ อัพรูปลงไป



The screenshot shows a code editor window with the following tabs at the top: index.js (selected), index.ejs, logo.png. The code editor displays the following JavaScript code:

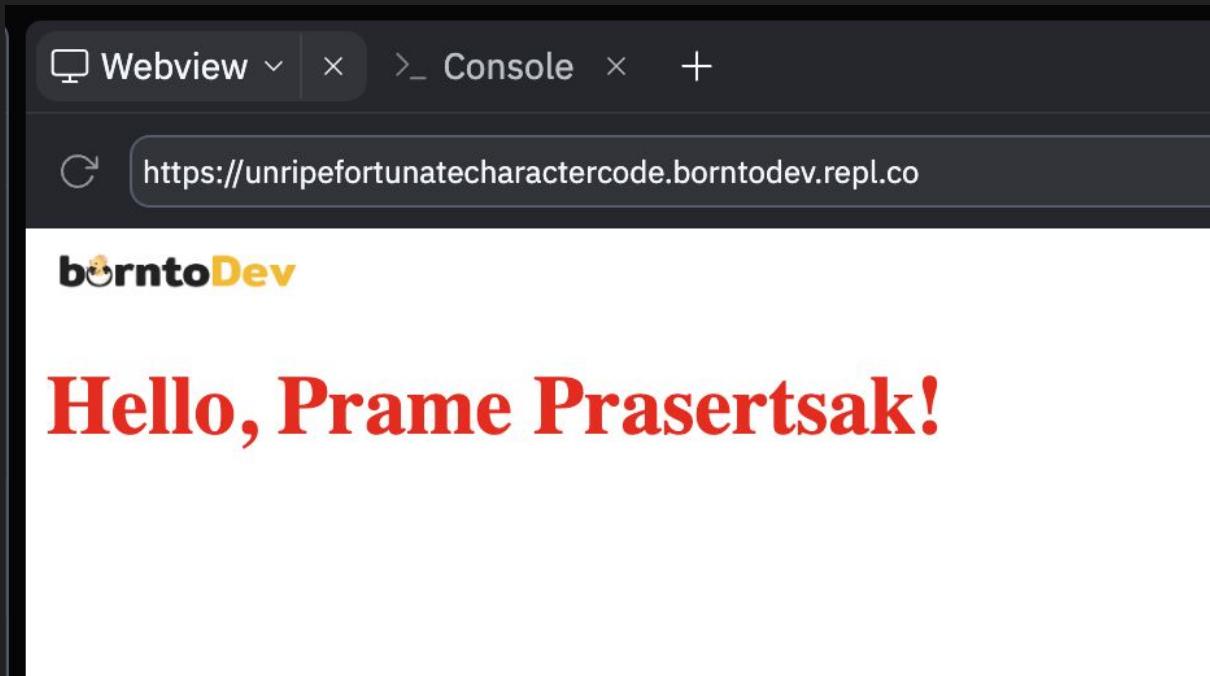
```
1 import express from 'express';
2
3 const app = express();
4 app.set('view engine', 'ejs');
5 app.use(express.static('public'));
6
7
```

The line `app.use(express.static('public'));` is highlighted with a brown background.

เพิ่มการใช้งาน static ลงใน index.js

```
index.js x index.ejs x logo.png x + :  
views > index.ejs  
1 <!DOCTYPE html>  
2 <html>  
3 <head>  
4   <title>Welcome</title>  
5 </head>  
6 <body>  
7     
8   <h1>Hello, <%= name %> <%= lastname %>!</h1>  
9 </body>  
10 </html>  
11
```

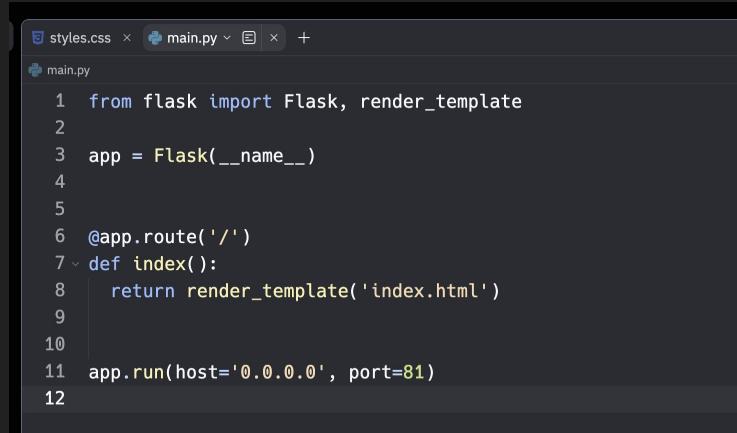
นำรูปภาพไปใส่ให้แสดงผลได้เลย



ลองทำกับ CSS ดูสิ !

```
19 v app.get('/items', (req, res) => {  
20     const items = ['Apple', 'Banana', 'Cherry'];  
21     res.render('items', { items });  
22});
```

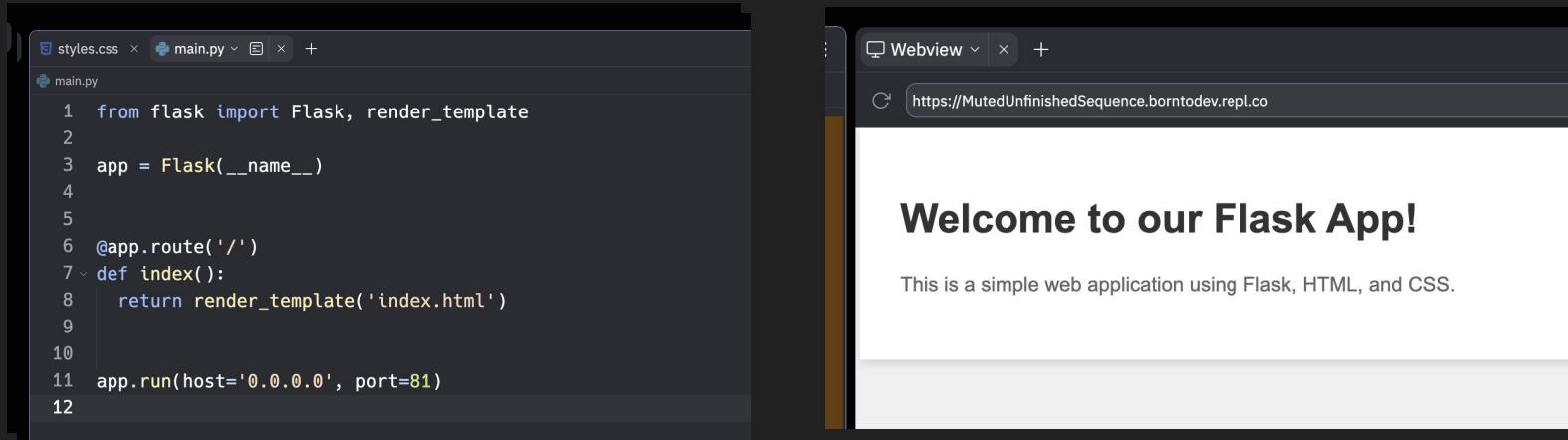
ส่งข้อมูลมาเป็น Array ก็ได้



The screenshot shows a code editor window with a dark theme. At the top, there are tabs for 'styles.css' and 'main.py'. The main area contains the following Python code:

```
from flask import Flask, render_template
app = Flask(__name__)
@app.route('/')
def index():
    return render_template('index.html')
app.run(host='0.0.0.0', port=81)
```

เสร็จแล้วกด Run ได้เลย



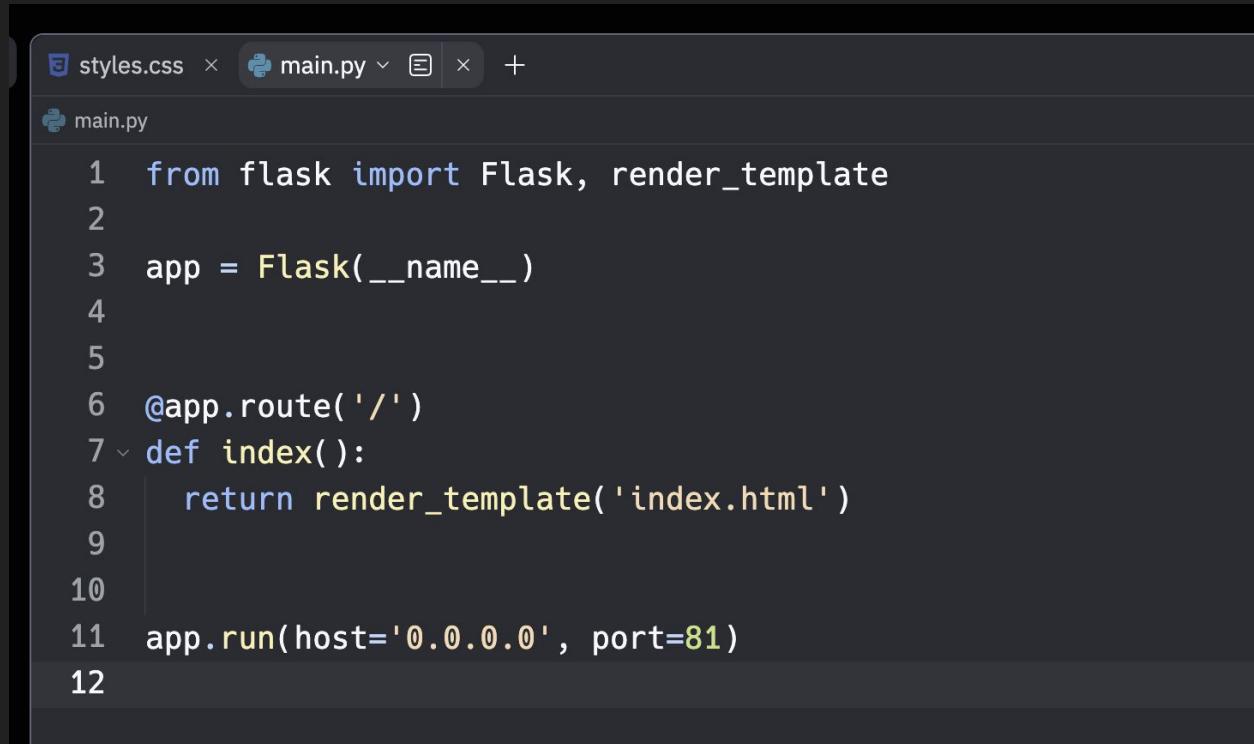
```
styles.css x main.py x +  
main.py  
1 from flask import Flask, render_template  
2  
3 app = Flask(__name__)  
4  
5  
6 @app.route('/')  
7 def index():  
8     return render_template('index.html')  
9  
10  
11 app.run(host='0.0.0.0', port=81)  
12
```

Webview x +  
<https://MutedUnfinishedSequence.borntodev.repl.co>

# Welcome to our Flask App!

This is a simple web application using Flask, HTML, and CSS.

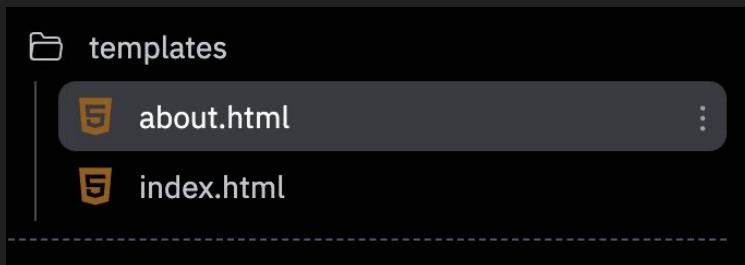
เสร็จแล้วกด Run ได้เลย



```
styles.css × main.py × +  
main.py  
1 from flask import Flask, render_template  
2  
3 app = Flask(__name__)  
4  
5  
6 @app.route('/')  
7 def index():  
8     return render_template('index.html')  
9  
10  
11 app.run(host='0.0.0.0', port=81)  
12
```

มาดูโค้ดตรงส่วนนี้กันดีกว่า

ถ้าเราต้องการ “เพิ่มหน้าใหม่” ลงในระบบ ?



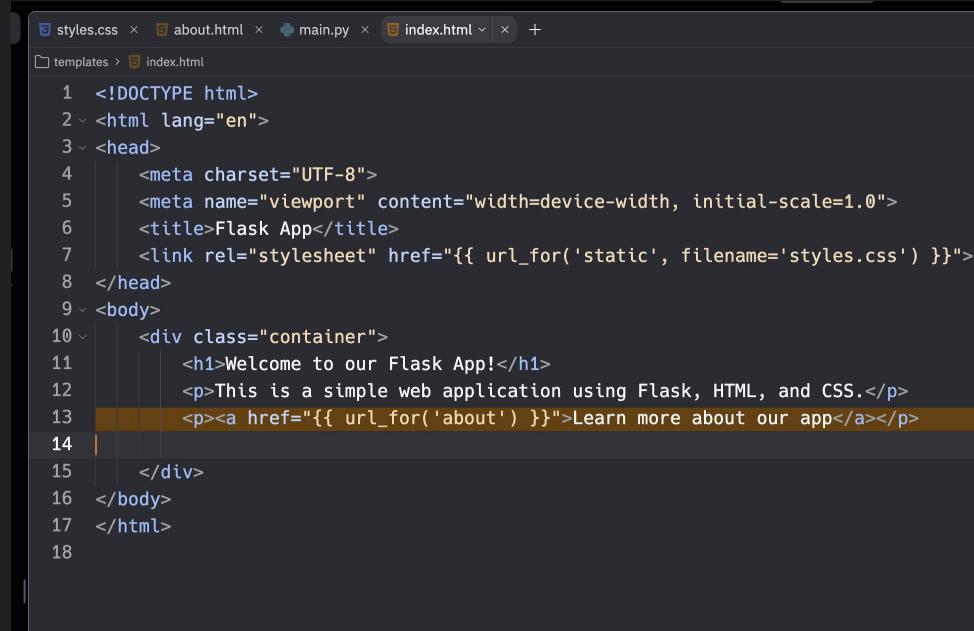
```
styles.css x about.html x +
templates > about.html
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">
5     <meta name="viewport" content="width=device-width, initial-scale=1.0">
6     <title>About - Flask App</title>
7     <link rel="stylesheet" href="{{ url_for('static', filename='styles.css') }}">
8 </head>
9 <body>
10    <div class="container">
11        <h1>About our Flask App</h1>
12        <p>This is the About page of our Flask web application. We're using Flask, HTML, and CSS to create a multi-page website.</p>
13        <p><a href="{{ url_for('home') }}>Go back to the Home page</a></p>
14    </div>
15 </body>
16 </html>
```

A screenshot of a code editor showing the content of the 'about.html' file. The code is written in HTML and includes meta tags for charset and viewport, a title, a link to a CSS file, and a main body section with a container div containing an h1 heading and a paragraph explaining the purpose of the page. The code editor has tabs for 'styles.css' and 'about.html' at the top, and a breadcrumb navigation bar showing 'templates > about.html'.

สร้างไฟล์ about.html ลงใน templates

```
1 from flask import Flask, render_template
2
3 app = Flask(__name__)
4
5
6 @app.route('/')
7 def index():
8     return render_template('index.html')
9
10 @app.route('/about')
11 def about():
12     return render_template('about.html')
13
14
15 app.run(host='0.0.0.0', port=81)
16
```

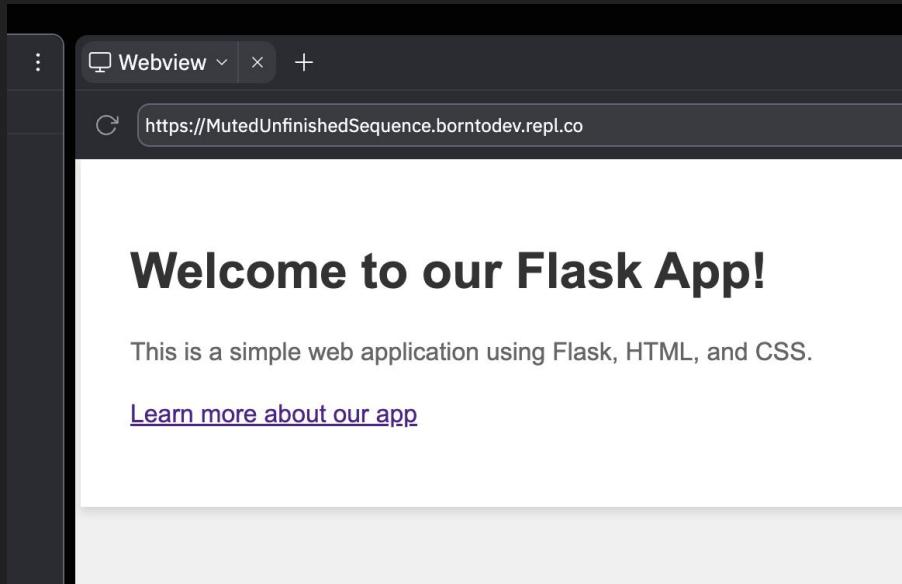
เพิ่ม @app.route ลงในส่วนของ about  
ใส่ก่อน app.run เท่านั้นนะ !



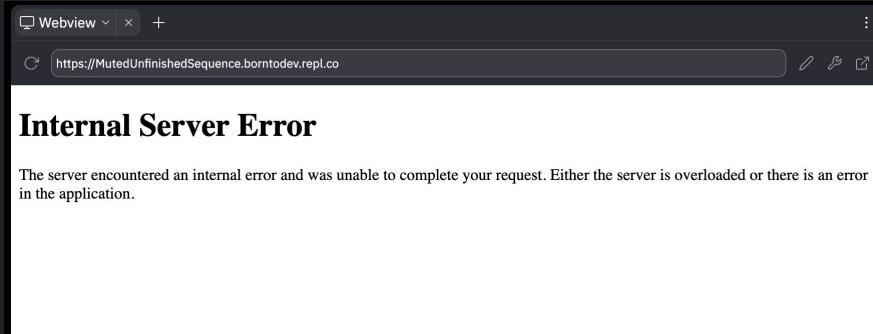
The screenshot shows a code editor with multiple tabs open at the top: styles.css, about.html, main.py, and index.html. The index.html tab is active, displaying the following HTML code:

```
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     <title>Flask App</title>
7     <link rel="stylesheet" href="{{ url_for('static', filename='styles.css') }}">
8 </head>
9 <body>
10    <div class="container">
11        <h1>Welcome to our Flask App!</h1>
12        <p>This is a simple web application using Flask, HTML, and CSS.</p>
13        <p><a href="{{ url_for('about') }}>Learn more about our app</a></p>
14    |
15    </div>
16 </body>
17 </html>
18
```

ສຸດທ້າຍເພີ່ມລົງກໍລົງໄປໃນ index.html  
ແລ້ວລອງ run ໄດ້ເລຍ !

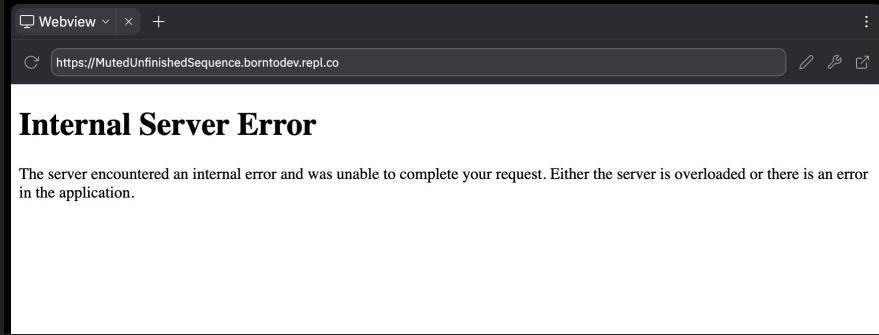


# ໂອເຄ ຂັ້ນໄວ້ ແລ້ວ ອຳນັດວ່າ ພັນຍາໄປ 50% ແລ້ວ



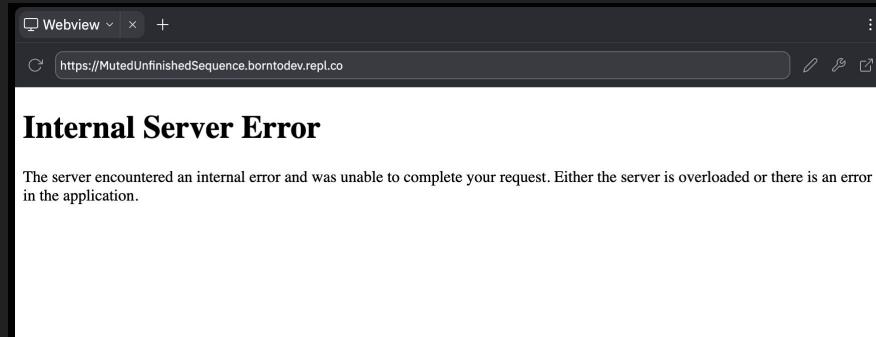
```
Console < x ④ Shell < +  
return self.ensure_sync(self.view_functions[rule.endpoint])(**view_args)  
File "main.py", line 12, in about  
    return render_template('about.html')  
File "/home/runner/MutedUnfinishedSequence/venv/lib/python3.10/site-packages/flask/template.py", line  
147, in render_template  
    return _render(app, template, context)  
File "/home/runner/MutedUnfinishedSequence/venv/lib/python3.10/site-packages/flask/template.py", line  
130, in _render  
    rv = template.render(context)  
File "/home/runner/MutedUnfinishedSequence/venv/lib/python3.10/site-packages/jinja2/environment.py", li  
ne 130, in render  
    self.environment.handle_exception()  
File "/home/runner/MutedUnfinishedSequence/venv/lib/python3.10/site-packages/jinja2/environment.py", li  
ne 936, in handle_exception  
    raise rewrite_traceback_stack(source=source)  
File "/home/runner/MutedUnfinishedSequence/templates/about.html", line 13, in top-level template code  
    <>><a href="{{ url_for('home') }}>Go back to the Home page</a></p>  
File "/home/runner/MutedUnfinishedSequence/venv/lib/python3.10/site-packages/flask/app.py", line 2031,  
in url_for  
    return self.handle_url_build_error(error, endpoint, values)  
File "/home/runner/MutedUnfinishedSequence/venv/lib/python3.10/site-packages/flask/app.py", line 2020,  
in url_for  
    rv = url_adapter.build( # type: ignoreUnionAttr )  
File "/home/runner/MutedUnfinishedSequence/venv/lib/python3.10/site-packages/werkzeug/routing/map.py",  
line 917, in build  
    raise BuildError(endpoint, values, method, self)  
werkzeug.routing.exceptions.BuildError: Could not build url for endpoint 'home'. Did you mean 'index' instead?  
172.31.128.1 - - [04/Apr/2023 07:42:05] "GET /about HTTP/1.1" 500 -
```

เมื่อคลิกไปที่ลิงก์จะพบว่ามัน .. **Error**



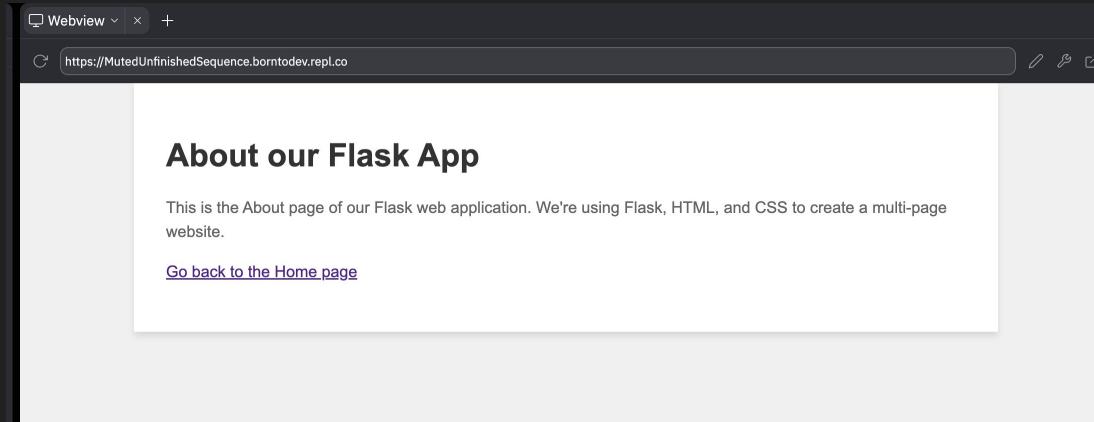
```
Console < x Shell +  
return self.ensure_sync(self.view_functions[rule.endpoint])(**view_args)  
File "/home/runner/MutedUnfinishedSequence/venv/lib/python3.10/site-packages/flask/template.py", line  
147, in render_template  
    return _render(app, template, context)  
File "/home/runner/MutedUnfinishedSequence/venv/lib/python3.10/site-packages/flask/template.py", line  
130, in _render  
    rv = template.render(context)  
File "/home/runner/MutedUnfinishedSequence/venv/lib/python3.10/site-packages/jinja2/environment.py", li  
ne 130, in render  
    self.environment.handle_exception()  
File "/home/runner/MutedUnfinishedSequence/venv/lib/python3.10/site-packages/jinja2/environment.py", li  
ne 936, in handle_exception  
    raise rewrite_traceback_stack(source=source)  
File "/home/runner/MutedUnfinishedSequence/templates/about.html", line 13, in top-level template code  
    <>><a href="{{ url_for('home') }}>Go back to the Home page</a></p>  
File "/home/runner/MutedUnfinishedSequence/venv/lib/python3.10/site-packages/flask/app.py", line 2031,  
in url_for  
    return self.handle_url_build_error(error, endpoint, values)  
File "/home/runner/MutedUnfinishedSequence/venv/lib/python3.10/site-packages/flask/app.py", line 2020,  
in url_for  
    rv = url_adapter.build( # type: ignoreUnionAttr )  
File "/home/runner/MutedUnfinishedSequence/venv/lib/python3.10/site-packages/werkzeug/routing/map.py",  
line 917, in build  
    raise BuildError(endpoint, values, method, self)  
werkzeug.routing.exceptions.BuildError: Could not build url for endpoint 'home'. Did you mean 'index' instead?  
172.31.128.1 -- [04/Apr/2023 07:42:05] "GET /about HTTP/1.1" 500 -
```

# Quiz : มาลองเดาดูกันว่าเกิดจากอะไร ?



```
File "/home/runner/mutedunfinisheSequence/venv/lib/python3.10/site-packages/werkzeug/routing/map.py", line 217, in
    raise BuildError(endpoint, values, method, self)
werkzeug.routing.exceptions.BuildError: Could not build url for endpoint 'home'. Did you mean 'index' instead?
172.31.128.1 - - [04/Apr/2023 07:42:05] "GET /about HTTP/1.1" 500 -
```

# Quiz : มาลองเดาดูกันว่าเกิดจากอะไร ?

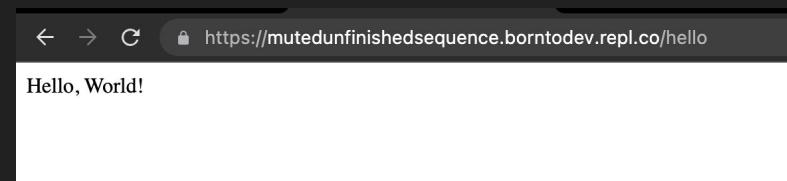


เมื่อแก้ไขเรียบร้อย หน้าเว็บก็จะออกมา  
ปกตินั่นเอง

คำถามที่อยากรีบลองเล่น  
“พอเราใช้ Template แล้ว อยากจะให้ return มาตรง ๆ  
เหมือนเดิม ยังทำได้อยู่ไหม ?”

```
1  from flask import Flask, render_template
2
3  app = Flask(__name__)
4
5
6  @app.route('/')
7  def index():
8      return render_template('index.html')
9
10 @app.route('/about')
11 def about():
12     return render_template('about.html')
13
14 @app.route('/hello')
15 def hello():
16     return "Hello, World!"
17
18
19 app.run(host='0.0.0.0', port=81)
20
```

```
1 from flask import Flask, render_template
2
3 app = Flask(__name__)
4
5
6 @app.route('/')
7 def index():
8     return render_template('index.html')
9
10 @app.route('/about')
11 def about():
12     return render_template('about.html')
13
14 @app.route('/hello')
15 def hello():
16     return "Hello, World!"
17
18
19 app.run(host='0.0.0.0', port=81)
20
```

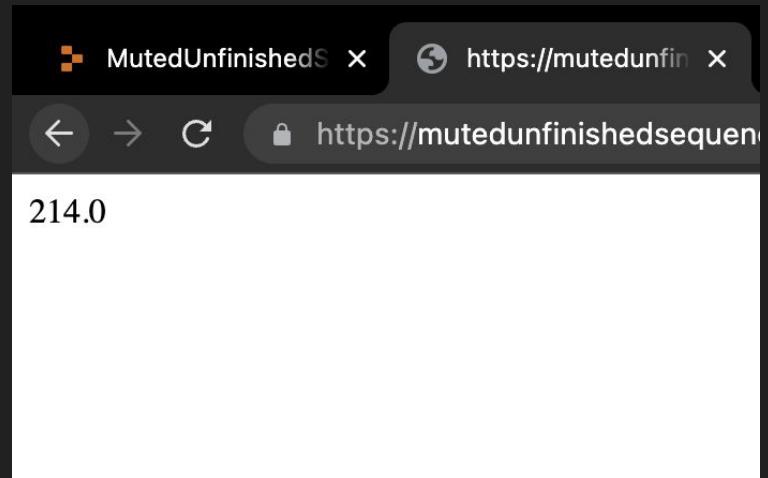




คำถามต่อมาที่อยากรู้คือคริส  
“พอ Flask มันใช้ Python แล้วเราเขียน  
Code ส่วน Python ปกติลงไปได้ใช่ไหม ?”

```
18 @app.route('/vatCal')
19 def vatCal():
20     return str(200*1.07)
```

```
18 @app.route('/vatCal')  
19 def vatCal():  
20     return str(200*1.07)
```



เราควรทำการประเมินผลนี้ ๆ  
ไว้ที่ back / front ?

# เราควรทำการประมวลผลนั้น ๆ ไว้ที่ back / front ?

## FE :

การตรวจสอบ Input, การตอบสนองกับหน้าตาแบบ real-time (responsive), การคำนวณประมวลผลทั่วไปที่ไม่จำเป็นต้องเก็บไว้ใน BE

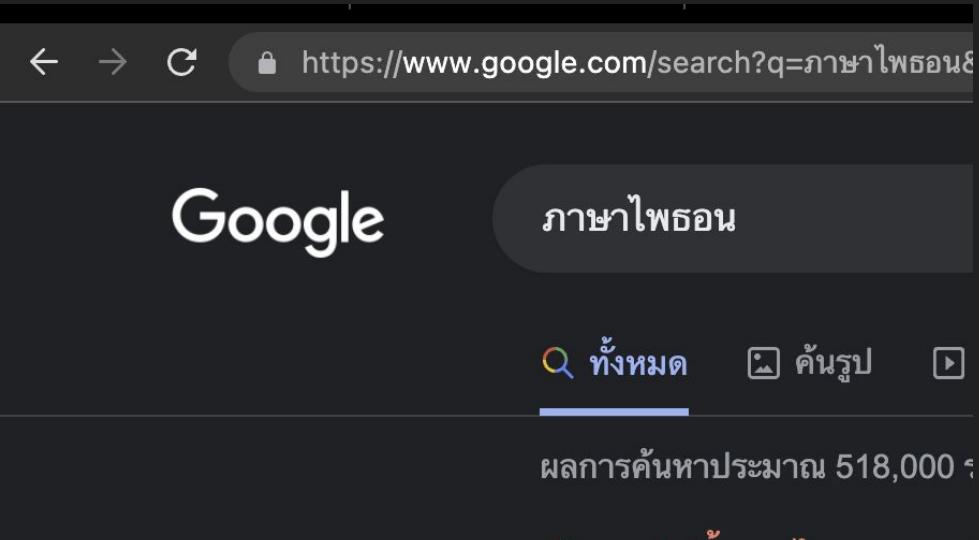
## BE :

การประมวลผลที่อาจจะต้องใช้การคำนวณที่ซับซ้อน สูตรเฉพาะ หรือ Algorithm ที่เปลี่ยนแปลง เชื่อมโยงข้อมูลจำนวนมาก, การประมวลผลข้อมูลที่มีความ Sensitive หรือ เป็นความลับ เช่น การยืนยันตัวตัว การเข้าสู่ระบบ

หลายครั้งเราจำเป้าที่จะต้องทำทั้งฝ่าย BE และ FE เพื่อการยืนยันความถูกต้อง

หากเราต้องการส่งค่าข้อมูลลงไปใน Page  
จะทำยังไงได้บ้าง ?

หากเราต้องการส่งค่าข้อมูลลงไปใน Page  
จะทำยังไงได้บ้าง ?



```
21
22 @app.route( '/greet/<name>' )
23 def greet(name):
24     return render_template('greeting.html', name=name)
25
26
```

อันดับแรก ให้เราทำการเพิ่มโค้ดส่วนนี้ไว้ใน  
Main.py

```
21
22 @app.route( '/greet/<name>' )
23 def greet(name):
24     return render_template('greeting.html', name=name)
25
26
```

โดยเราจะส่ง Param เข้าไปผ่านทาง <name>

```
  @app.route('/about')
  def about():
      return render_template('about.html')
```

```
21
22 @app.route('/greet/<name>')
23 def greet(name):
24     return render_template('greeting.html', name=name)
25
26
```

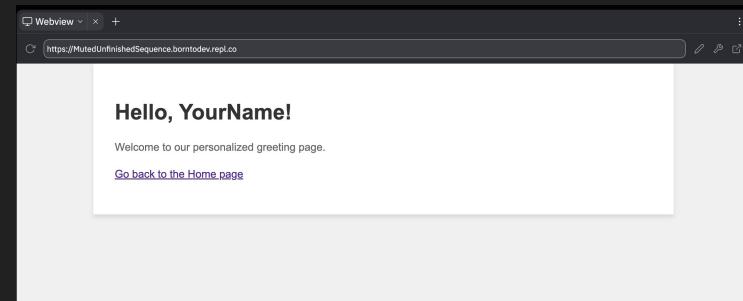
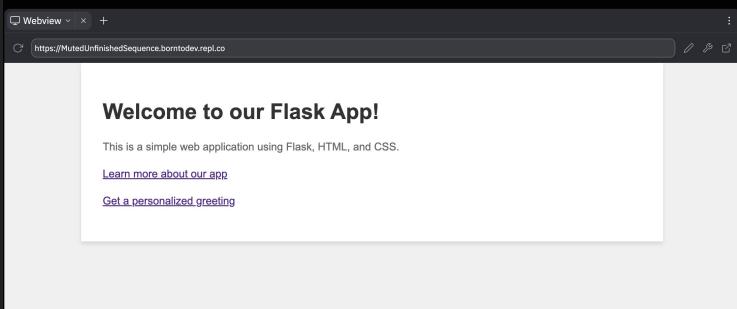
โดยเราจะส่ง Param เข้าไปผ่านทาง <name>  
อย่าลืม ! กำหนดข้อมูลด้านหลังด้วย

```
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     <title>Greeting - Flask App</title>
7     <link rel="stylesheet" href="{{ url_for('static',
filename='styles.css') }}">
8 </head>
9 <body>
10 <div class="container">
11     <h1>Hello, {{ name }}!</h1>
12     <p>Welcome to our personalized greeting page.</p>
13     <p><a href="{{ url_for('index') }}>Go back to the Home
page</a></p>
14 </div>
15 </body>
16 </html>
```

ต่อมา ทำการสร้างหน้า greeting.html

```
14      <p><a href="{{ url_for('greet', name='YourName') }}>Get a  
15          personalized greeting</a></p>
```

ສຸດທ້າຍ ວັດທະນາໄລໃນໜ້າ index.html

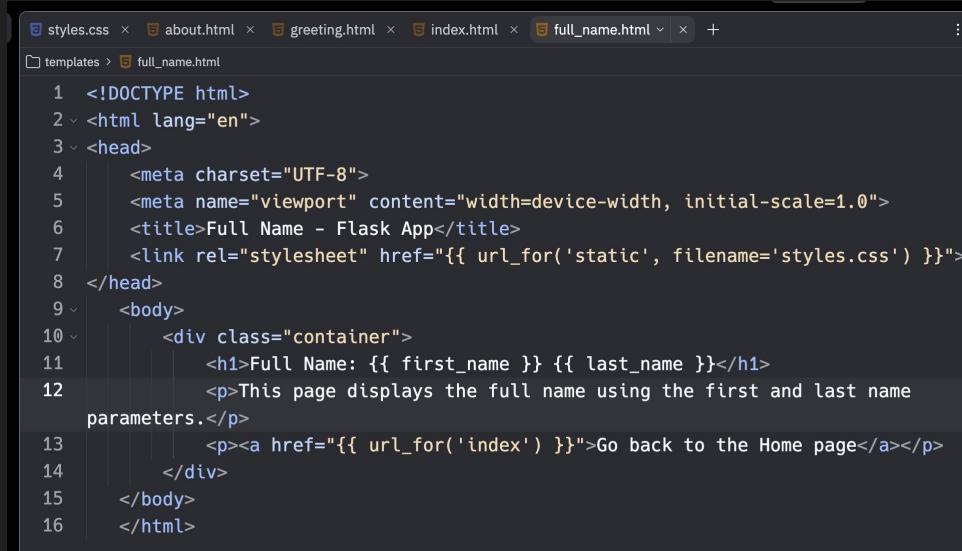


ทดสอบกันได้เลย !

ไม่พอใจ ? อยากส่งมากกว่านี้ ?

```
26 @app.route('/fullname/<first_name>/<last_name>')
27 def fullname(first_name, last_name):
28     return render_template('full_name.html', first_name=first_name,
29                           last_name=last_name)
```

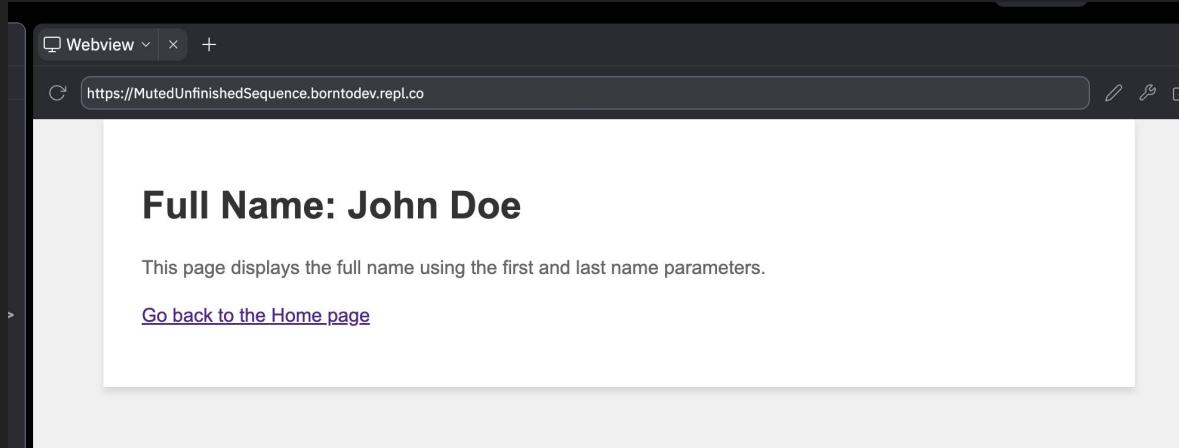
# ดำเนินการเพิ่ม app.route ใหม่ลงไป



```
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      <title>Full Name - Flask App</title>
7      <link rel="stylesheet" href="{{ url_for('static', filename='styles.css') }}>
8  </head>
9  <body>
10     <div class="container">
11         <h1>Full Name: {{ first_name }} {{ last_name }}</h1>
12         <p>This page displays the full name using the first and last name
13             parameters.</p>
14         <p><a href="{{ url_for('index') }}>Go back to the Home page</a></p>
15     </div>
16 </body>
17 </html>
```

# สร้างไฟล์ full\_name.html ใน templates

```
'or('fullname', first_name='John', last_name='Doe' ) {}  
'a></p>
```



มาทดสอบกันได้เลย

# พามา Workshop (กับเวลาที่เหลืออยู่อันน้อยนิด)

“ให้ลองสร้างหน้าใหม่ชื่อ bmiCalculate โดยรับตัวเลขมาทั้งหมด 2 ตัว แล้วทำการรับตัวเลขนำหนัก และ ส่วนสูง แล้วส่งตัวเลขที่คำนวณได้ ออกมายังหน้าเว็บของเรา”

# พามา Workshop (กับเวลาที่เหลืออยู่อันน้อยนิด)

“ให้ลองสร้างหน้าใหม่ชื่อ bmiCalculate โดยรับ ตัวเลขมาทั้งหมด 2 ตัว และทำการรับตัวเลข น้ำหนัก และ ส่วนสูง และ ส่งตัวเลขที่คำนวณได้ ออกมานหน้าเว็บของเรา”



The screenshot shows a code editor window with a dark theme. At the top, there are tabs for 'Console', 'Shell', and 'main.py'. The main area displays the following Python code:

```
30 @app.route('/bmi/<float:height>/<float:weight>')
31 def bmi(height, weight):
32     # Calculate BMI
33     bmi_value = round(weight / (height / 100) ** 2, 2)
34
35     return render_template('bmi.html', height=height, weight=weight, bmi_value=bmi_value)
36
37
```

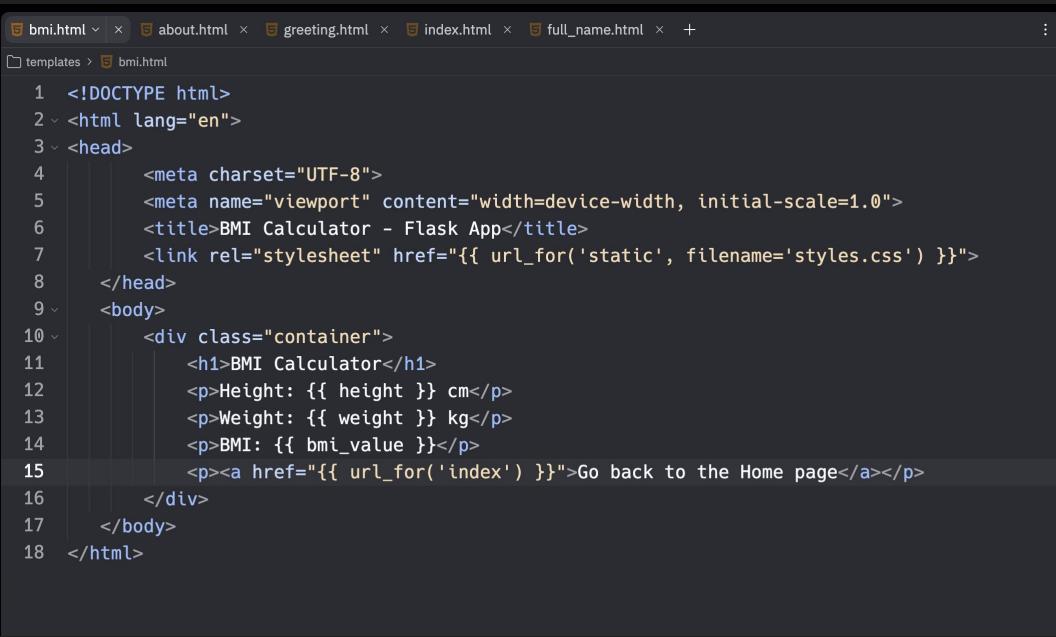
# เพิ่ม app.route การรับข้อมูล ลงไป



The screenshot shows a terminal window with tabs for 'Console', 'Shell', and 'main.py'. The main tab displays the following Python code:

```
30 @app.route('/bmi/<float:height>/<float:weight>')
31 def bmi(height, weight):
32     # Calculate BMI
33     bmi_value = round(weight / (height / 100) ** 2, 2)
34
35     return render_template('bmi.html', height=height, weight=weight, bmi_value=bmi_value)
36
37
```

เพิ่ม app.route การรับข้อมูล ลงไป  
สังเกตได้ว่าเราสามารถระบุประเภทข้อมูลที่จะโอนเข้าไป  
ได้ !



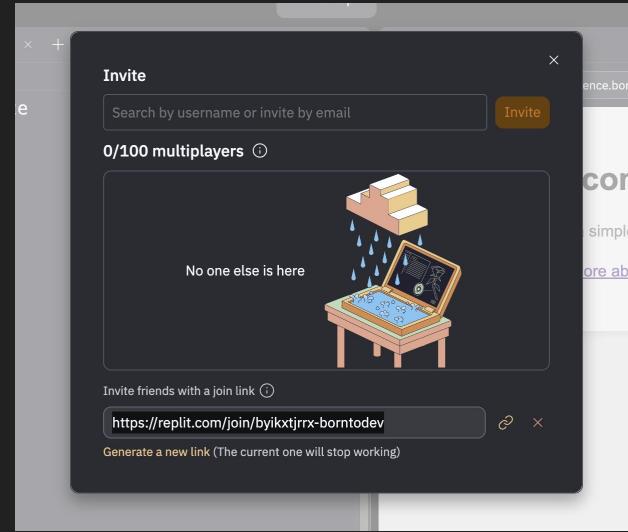
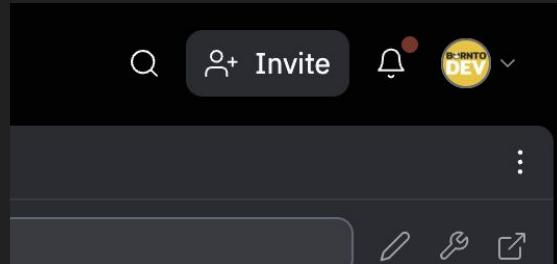
```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">
5     <meta name="viewport" content="width=device-width, initial-scale=1.0">
6     <title>BMI Calculator - Flask App</title>
7     <link rel="stylesheet" href="{{ url_for('static', filename='styles.css') }}>
8 </head>
9 <body>
10 <div class="container">
11     <h1>BMI Calculator</h1>
12     <p>Height: {{ height }} cm</p>
13     <p>Weight: {{ weight }} kg</p>
14     <p>BMI: {{ bmi_value }}</p>
15     <p><a href="{{ url_for('index') }}>Go back to the Home page</a></p>
16 </div>
17 </body>
18 </html>
```

สร้างไฟล์ bmi.html เพื่อรับข้อมูลเข้ามา

```
17
18      <p><a href="{{ url_for('bmi', height=170, weight=70) }}">Calculate example BMI</a></p>
19
```

สุดท้ายเพิ่มลิงก์ลงไปใน index.html

# การส่งการบ้านด้วย Repl.it



ให้สร้างลิงก์จากปุ่ม **Invite** และ กด **Generate Link** แล้ว  
ดำเนินการคัดลอกมาใส่ใน ตอบคำถาวม

# การส่งการบ้านด้วย Repl.it

## BMI Calculator

Height: 170.0 cm

Weight: 70.0 kg

BMI: 24.22

[Go back to the Home page](#)

Result : [คุณปกติ / คุณอ้วนระดับ 1 / ... ]

เพิ่มข้อความผลลัพธ์ของ BMI ออกมานหน้าจอ