Web App Development

DAY 9 - ROAD TO FRONT-END DEVELOPER BOOTCAMP

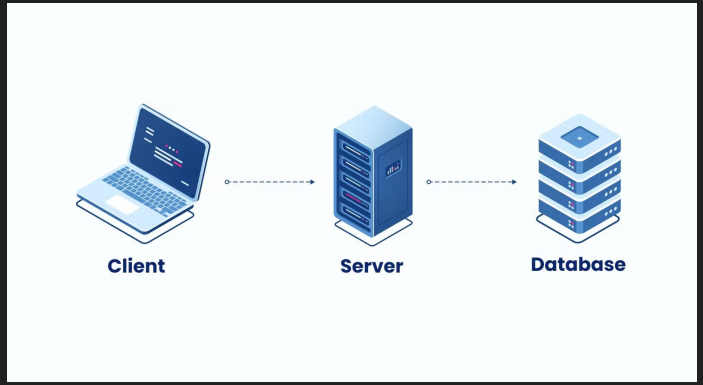
เราเรียนอะไรกันไปบ้าง ?

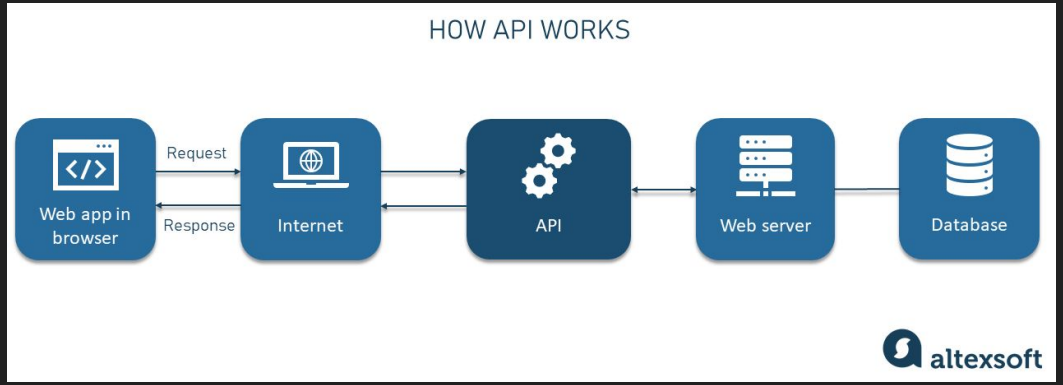
HTML CSS JavaScript

FE : HTML CSS JavaScript

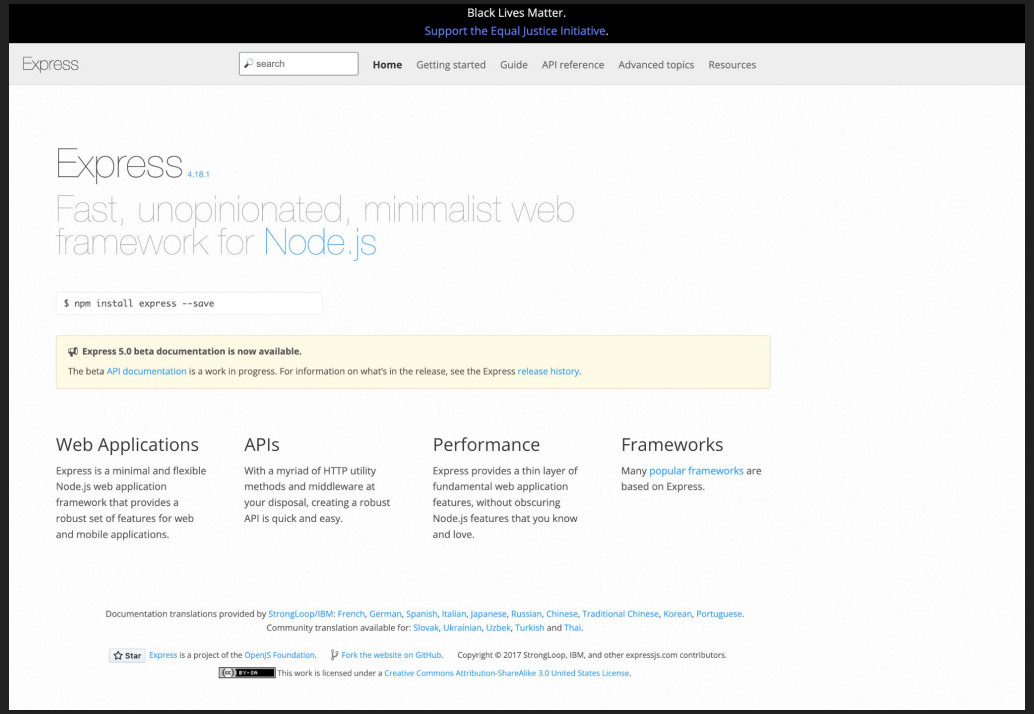
แล้ว Back-end หละ เขาทำอะไร ?

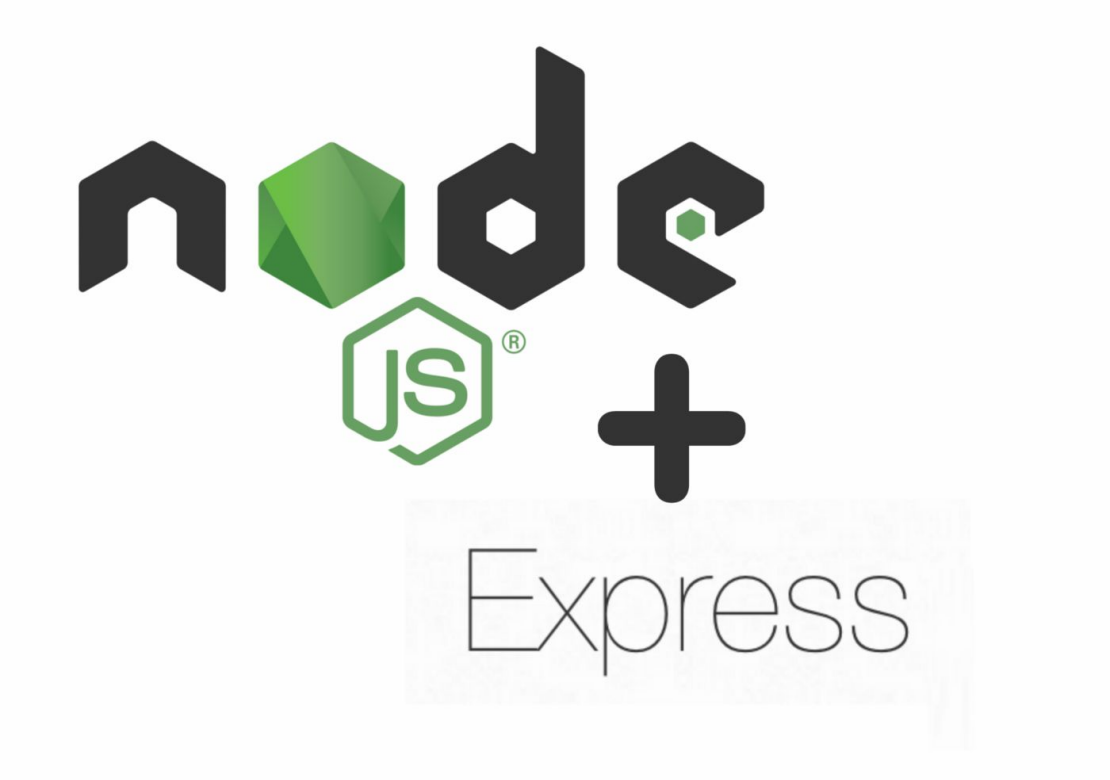
แล้ว Back-end หละ เขาทำอะไร ?

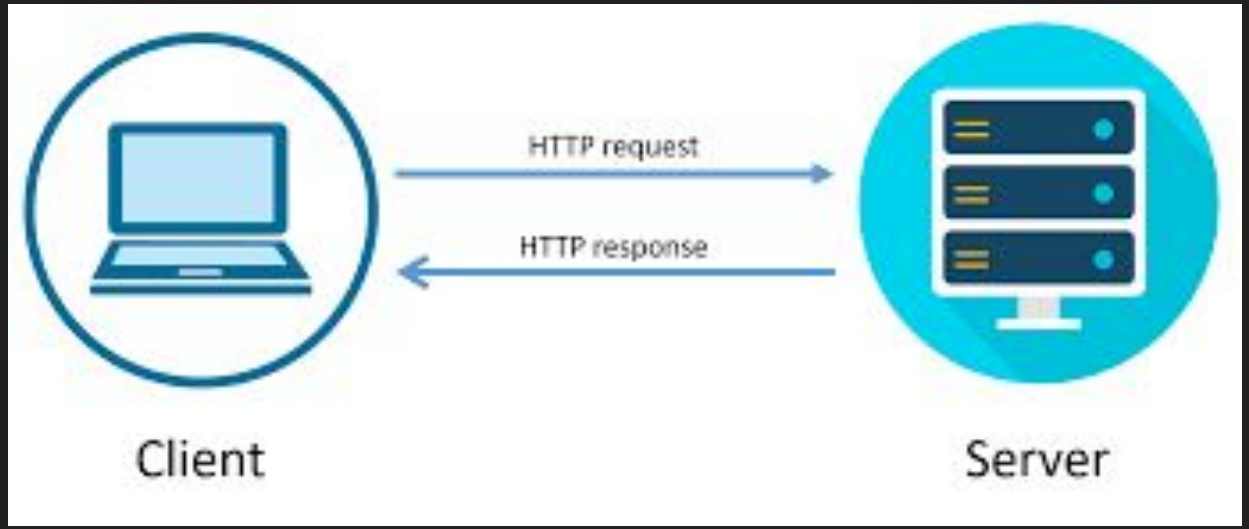




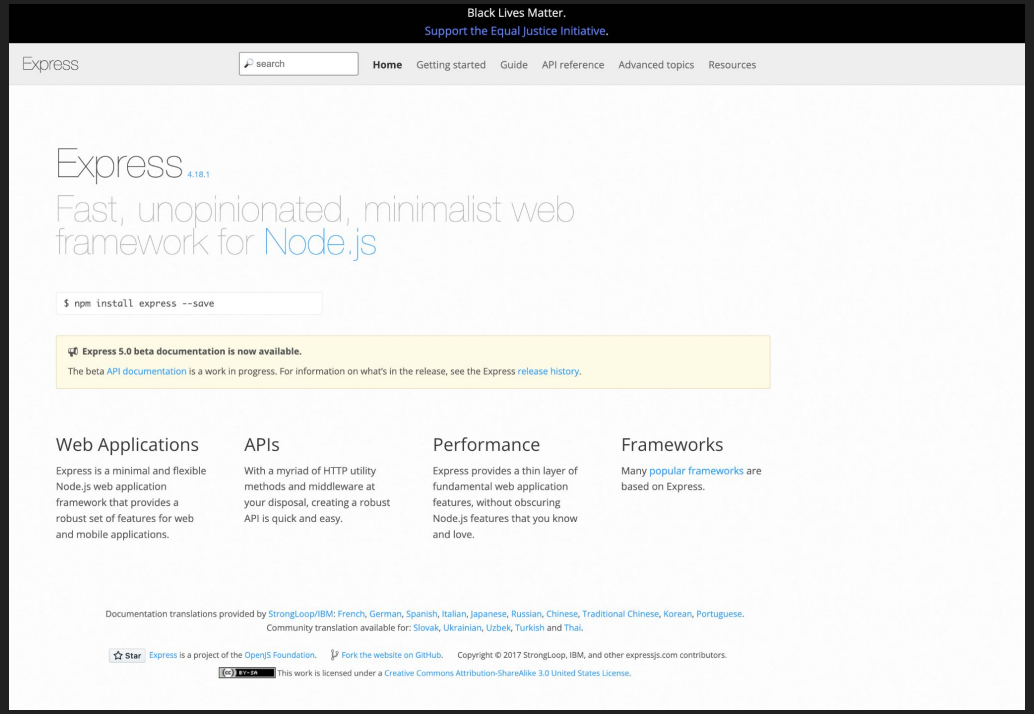
มารู้จักกับ Express




node
JS
+
Express


HTTP request
HTTP response
Client
Server

มารู้จักกับ Express



ฟีเจอร์หลัก

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

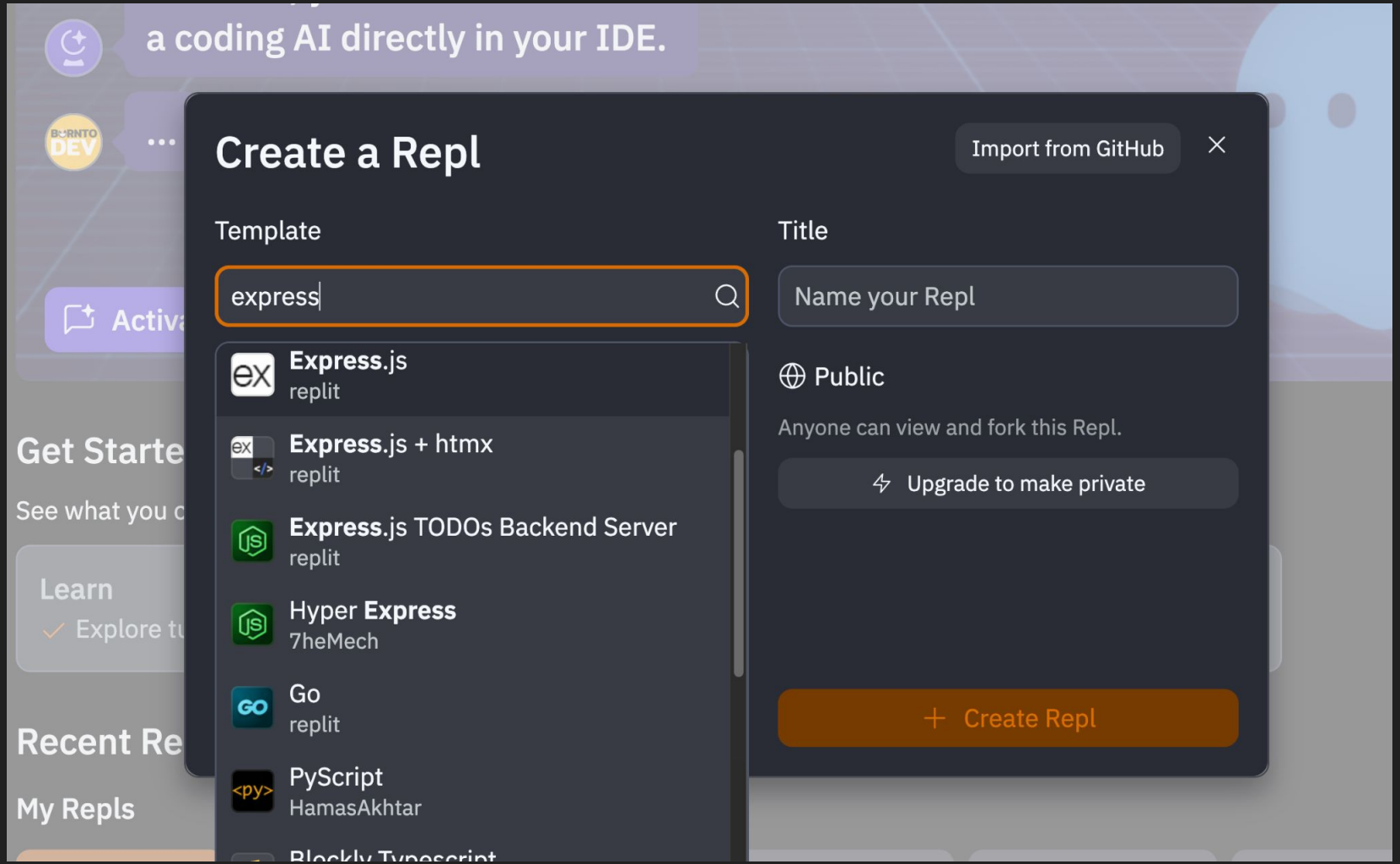
APIs: Build RESTful APIs for various client-side applications.

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

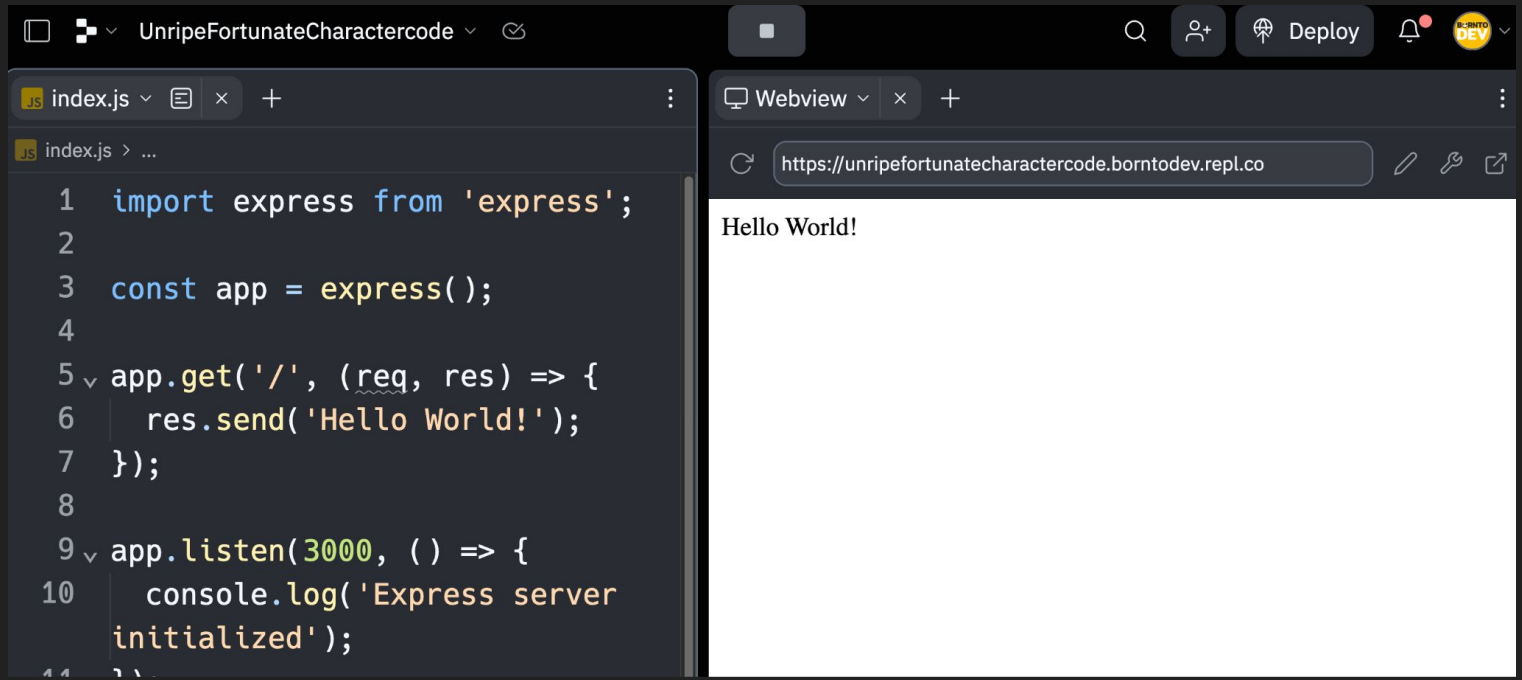
Streaming Services: Handle real-time data streaming.

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
Title
Name your Repl
Public
Anyone can view and fork this Repl.
Upgrade to make private
Create Repl
Get Starte
See what you c
Learn
Explore tu
Recent Re
My Repls

เมื่อเรียบร้อยให้กดปุ่ม Run



เราจะพบว่าหน้าเว็บของเราปรากฏขึ้นแล้ว

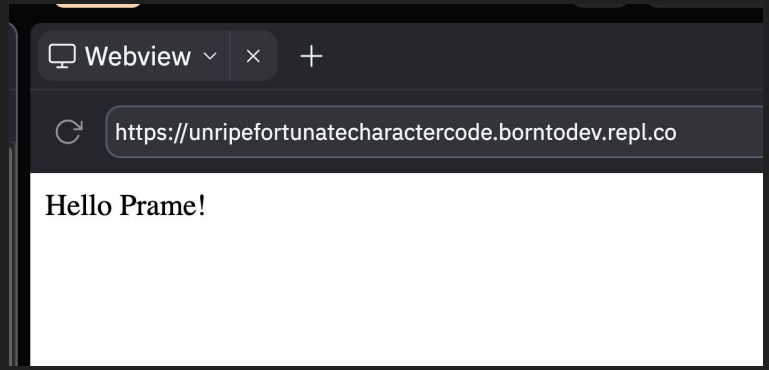
```python
@app.route('/')
def index():
    return 'Hello from Flask!'
```

ทดลองเปลี่ยนข้อความตรงนี้ดูกัน !

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

ทดลองเปลี่ยนข้อความตรงนี้ดูกัน !

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

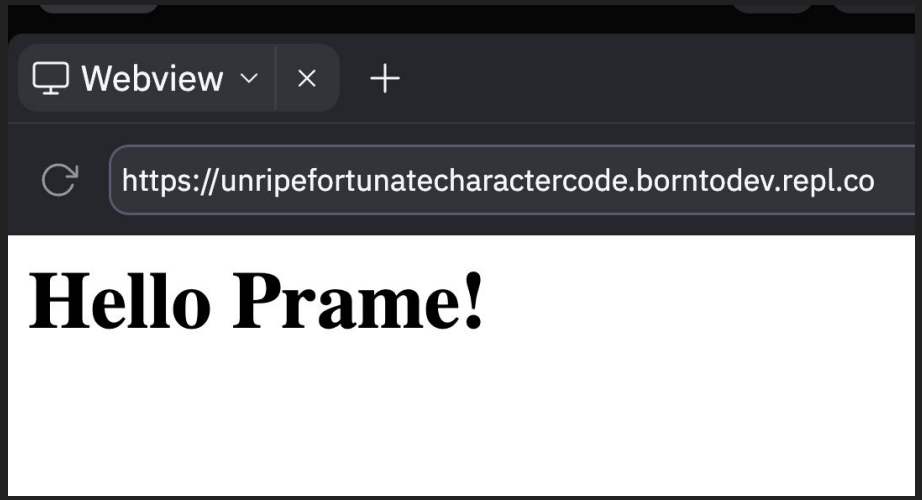


สังเกตดูว่าข้อความจะถูกเปลี่ยนไปแล้ว

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

ทดลองใส่ Tag HTML ดูกัน !

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



เรียบร้อยยย !

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

ว่าแต่ app.get อะไรนี่คืออะไรกัน ?

งั้นแปลว่าเราเอาทั้งเว็บใส่ไปในนี้
ก็จะได้เลยใช่ไหม ?

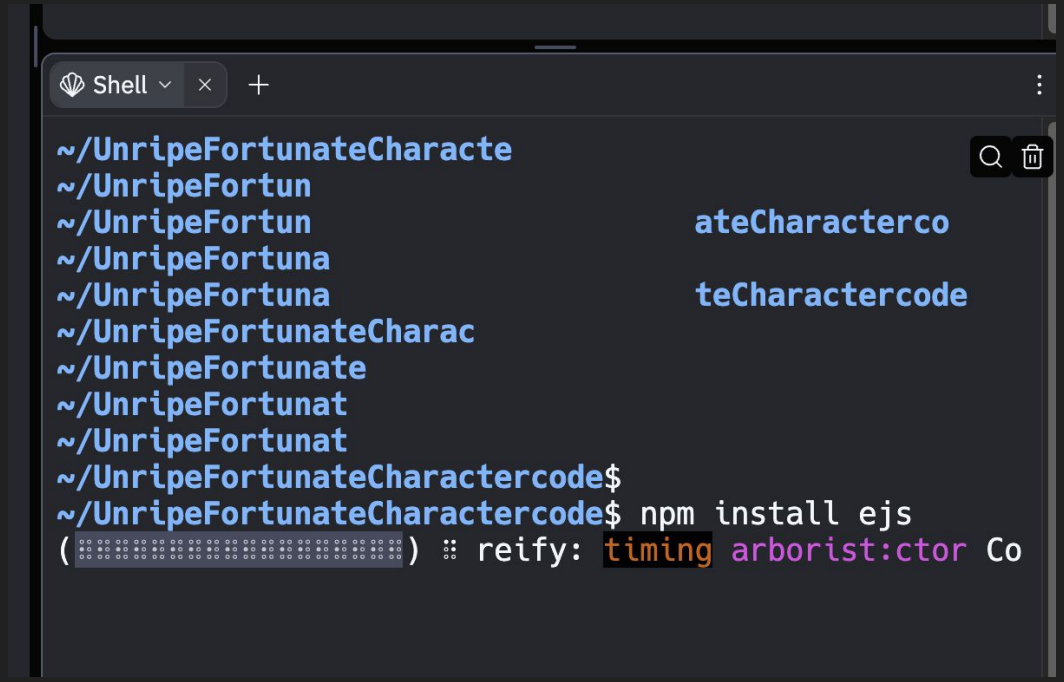
```
5 app.get('/', (req, res) => {
6     res.send('<h1>Hello Prame!</h1>');
7 });
```

งั้นแปลว่าเราเอาทั้งเว็บใส่ไปในนี้
ก็จะได้เลยใช่ไหม ?

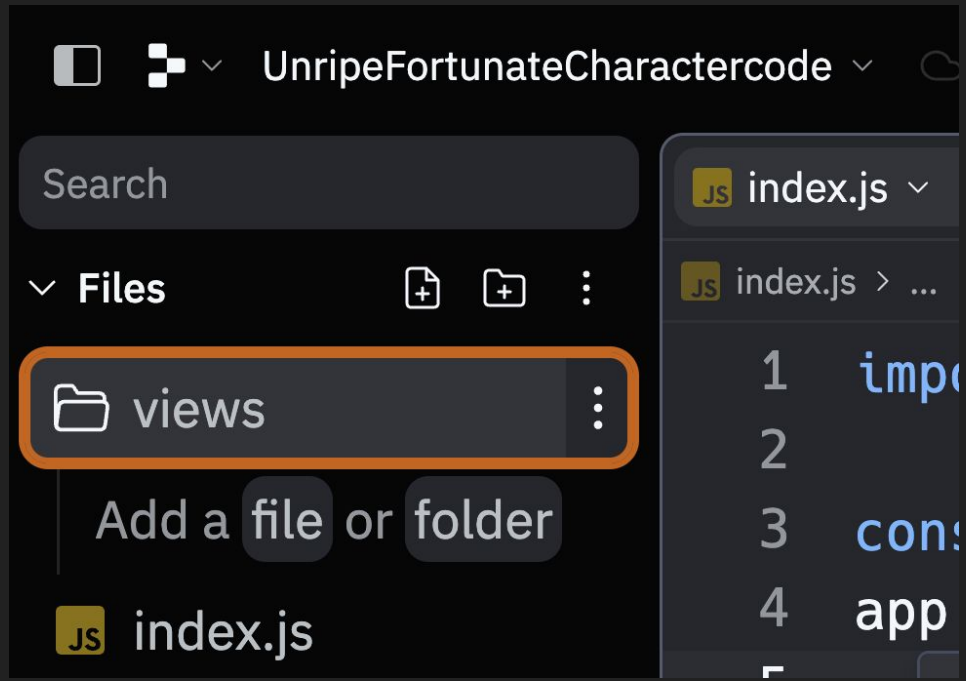
คำตอบคือ ใช่ ! แต่มันมีวิธีที่ดีกว่านี้เยอะอยู่ !

เราเรียกสิ่งนี้ว่าการใช้งาน

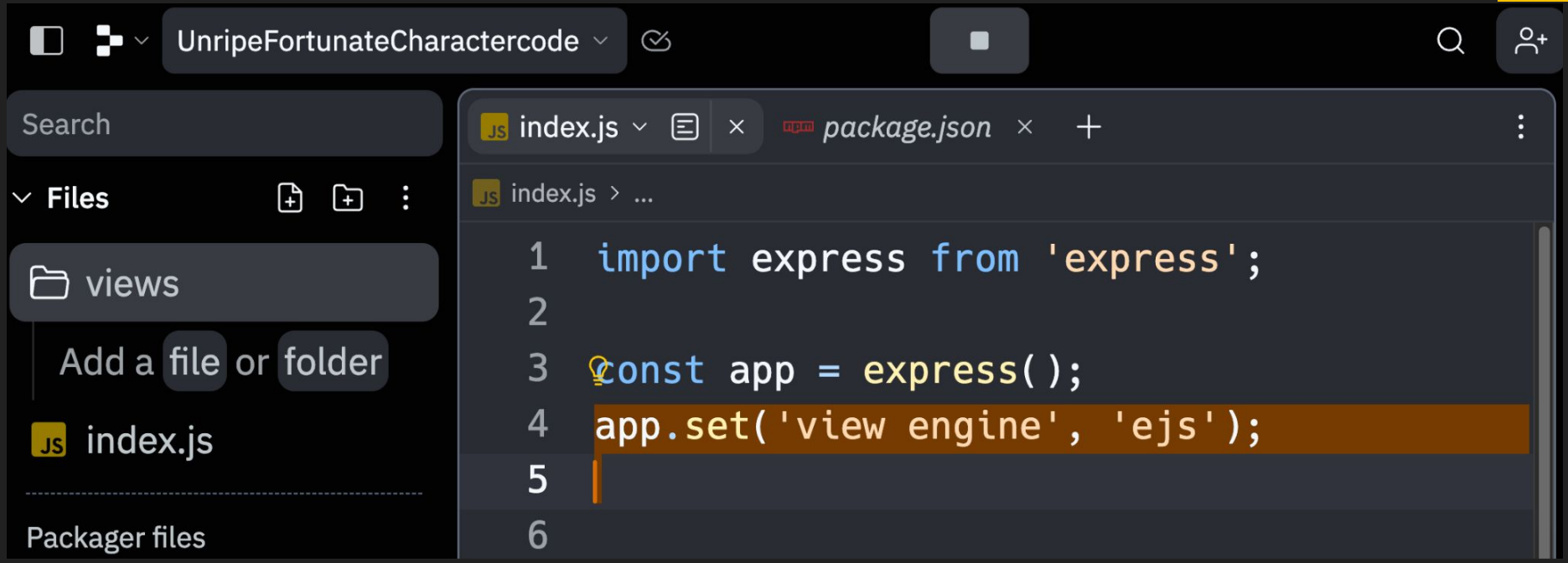
Template Engine



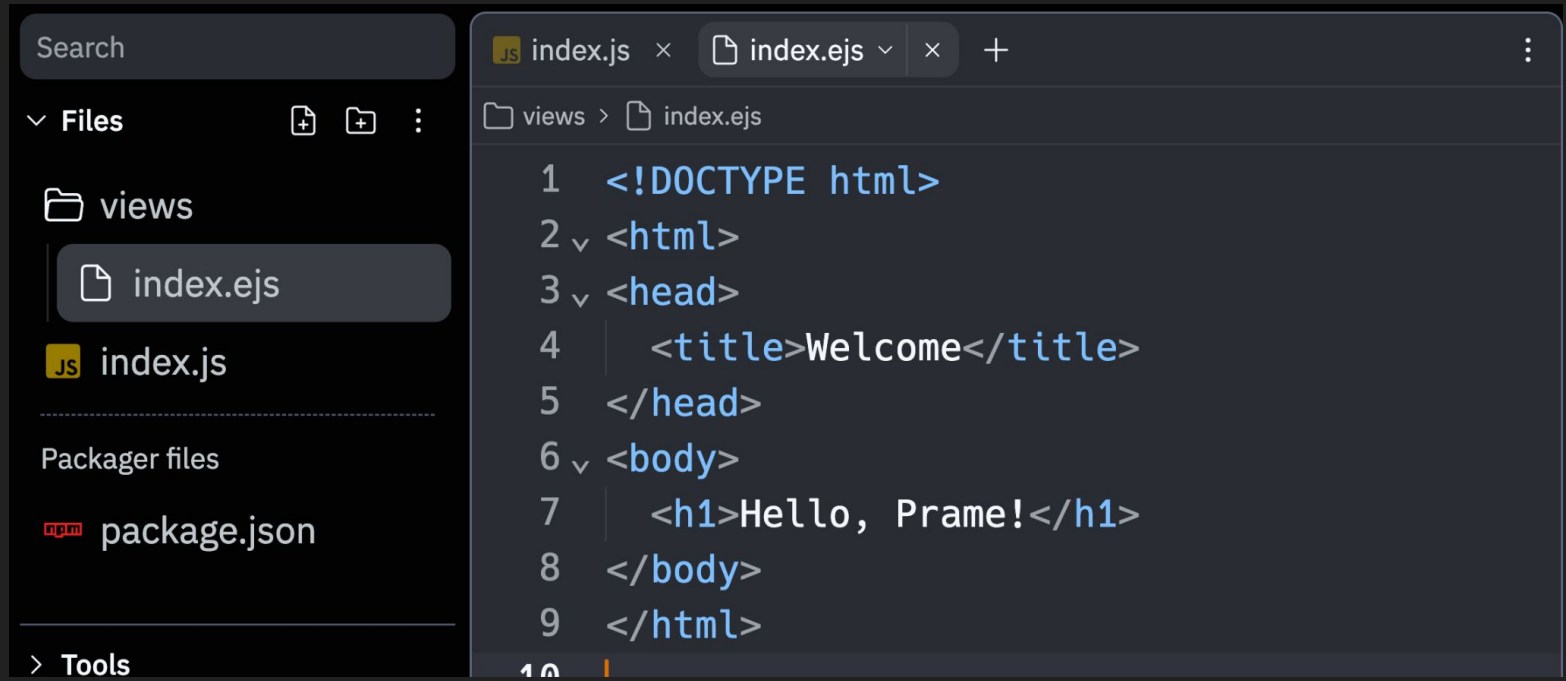
ติดตั้ง ejs ผ่าน shell ด้วยคำสั่ง npm install ejs



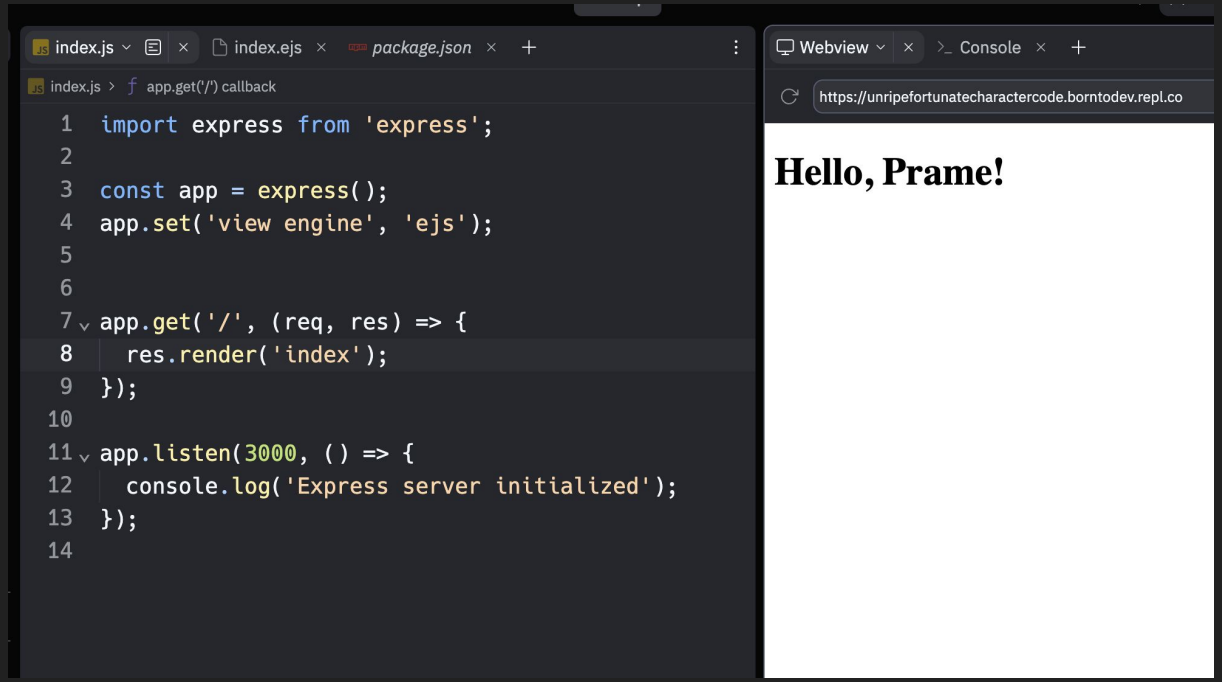
สร้างโฟลเดอร์ views ขึ้นมา



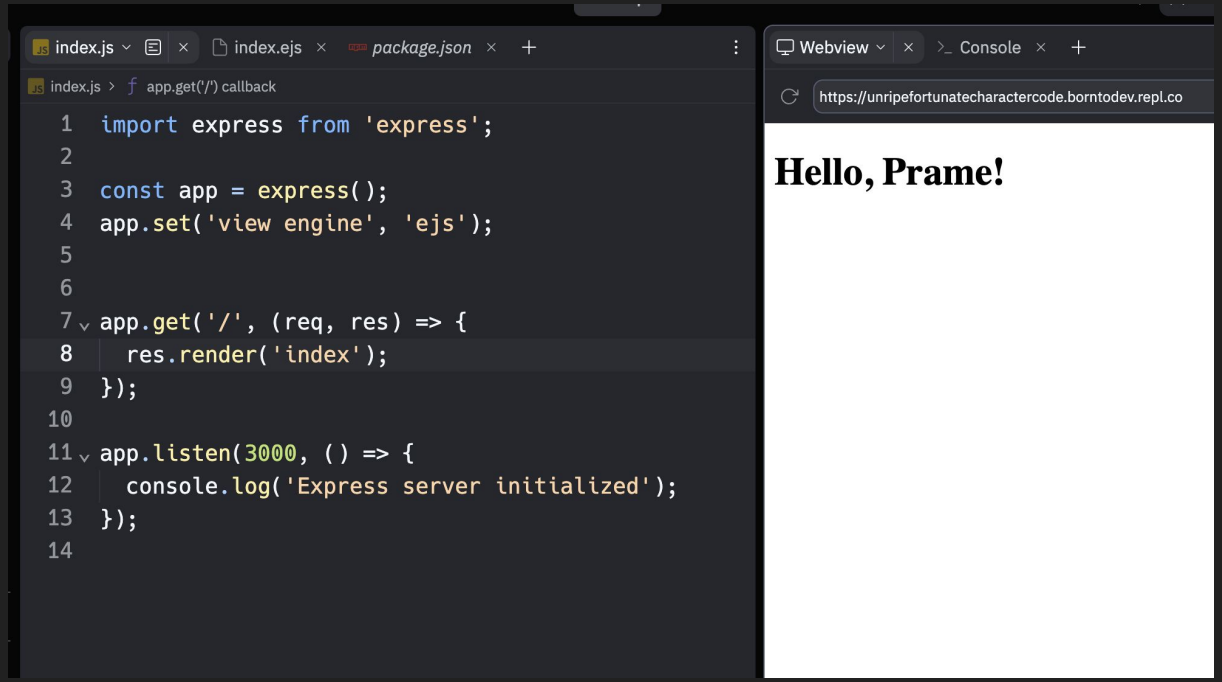
เรียกใช้ view engine กับ ejs



สร้างไฟล์ index.ejs ไว้ใน views



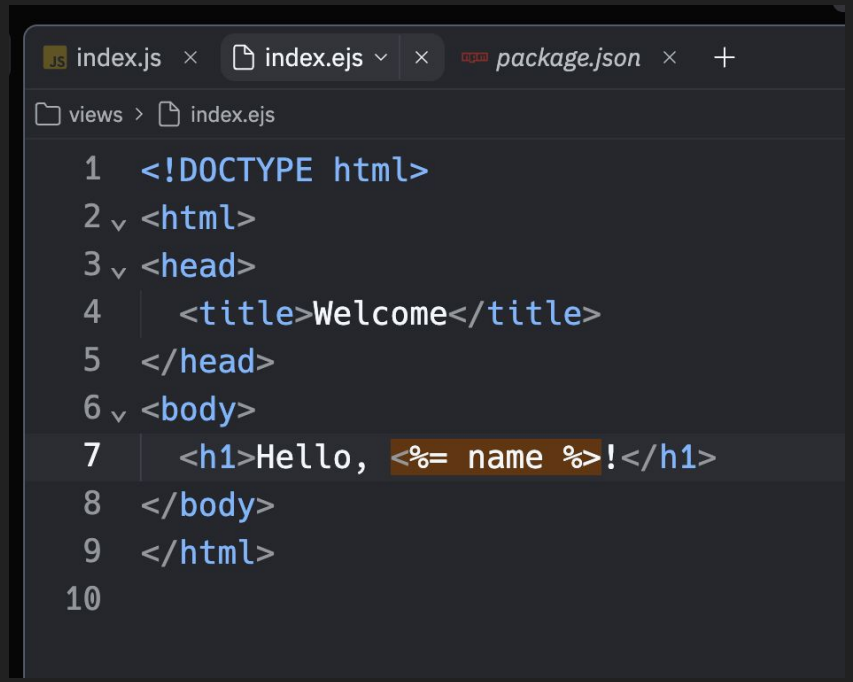
ปรับแก้ app.get(‘/’) เป็น render



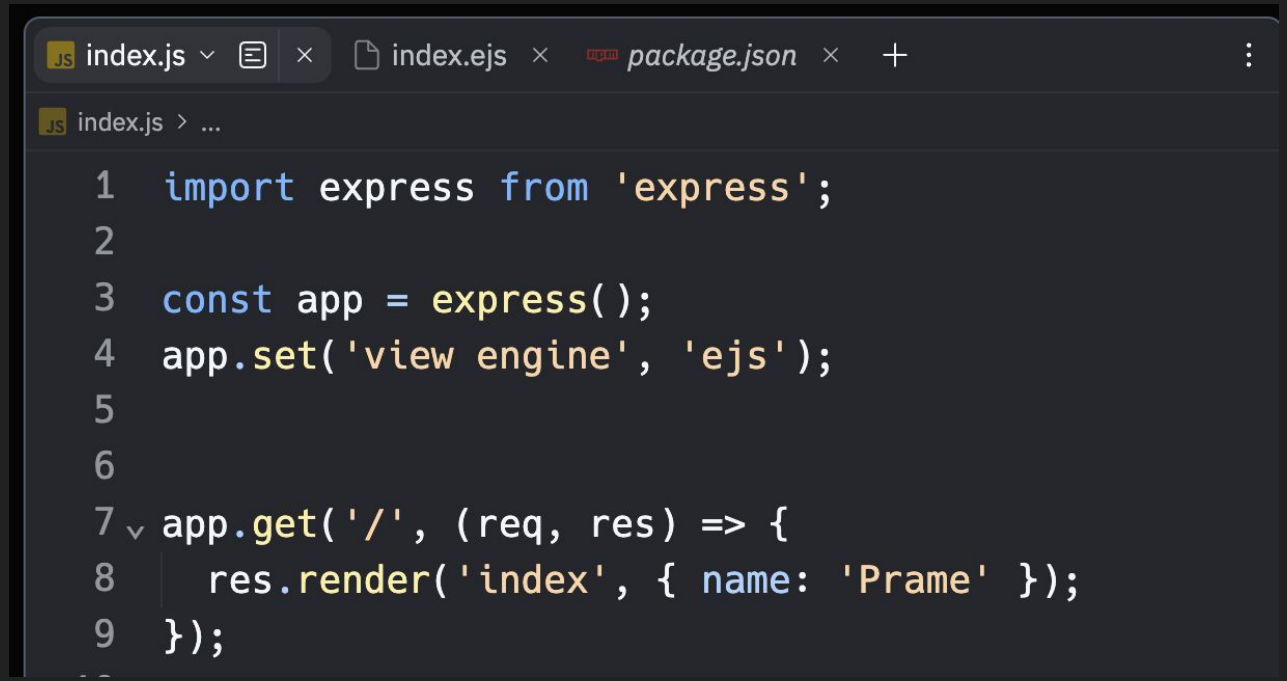
เว็บก็แสดงแล้ว

Template Engine มีเอาไว้ทำ

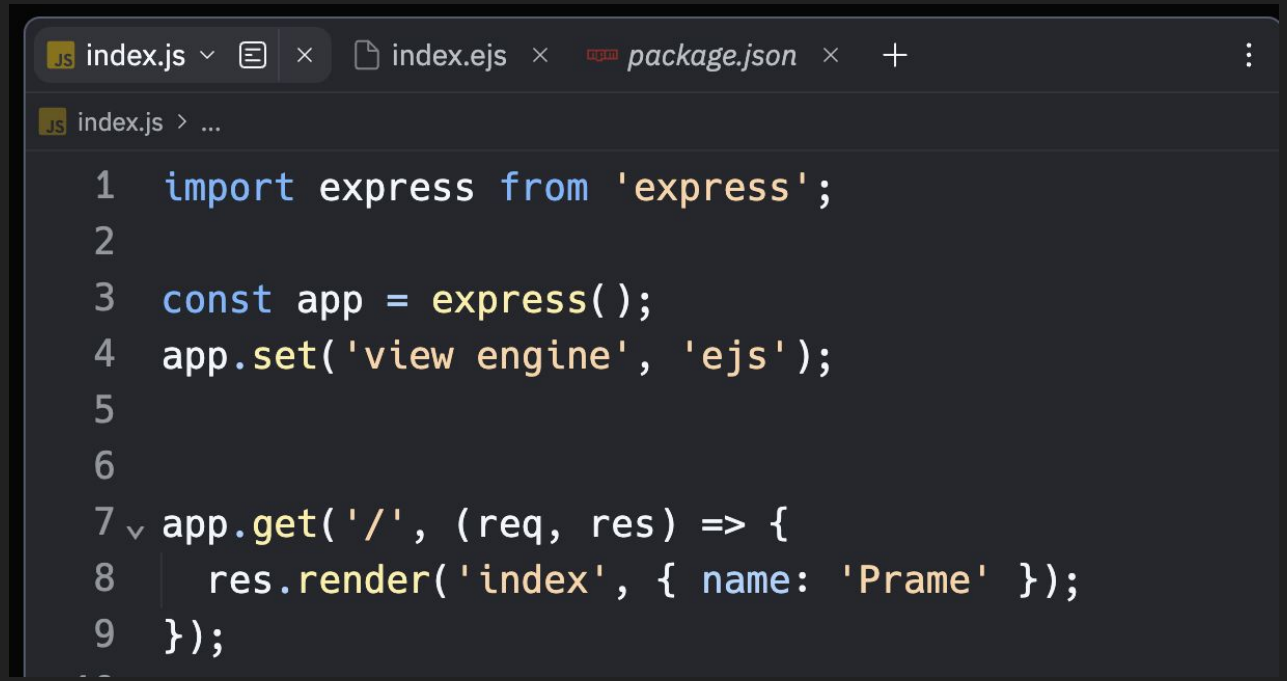
Dynamic Data ลงในไฟล์ HTML ของเรา



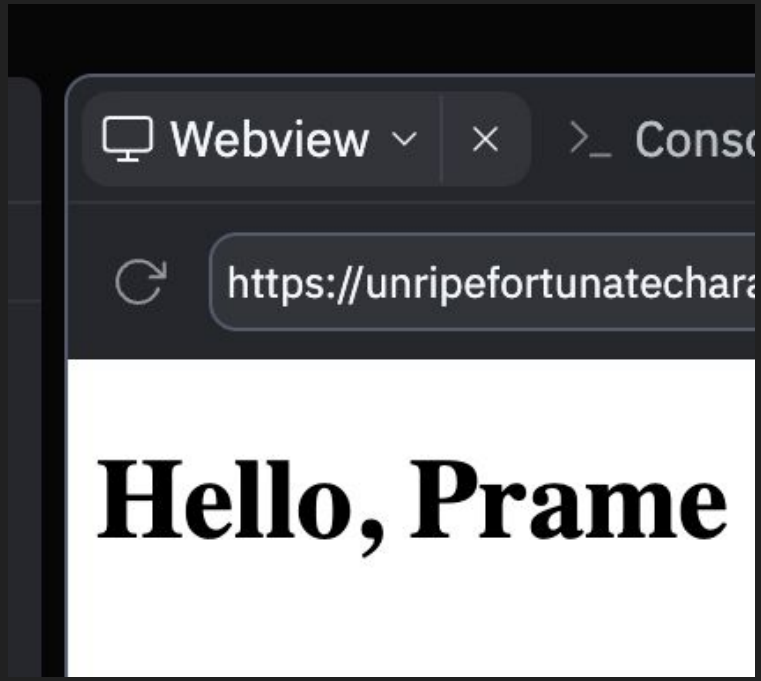
ลองเพิ่ม โค้ดนี้ ลงไป



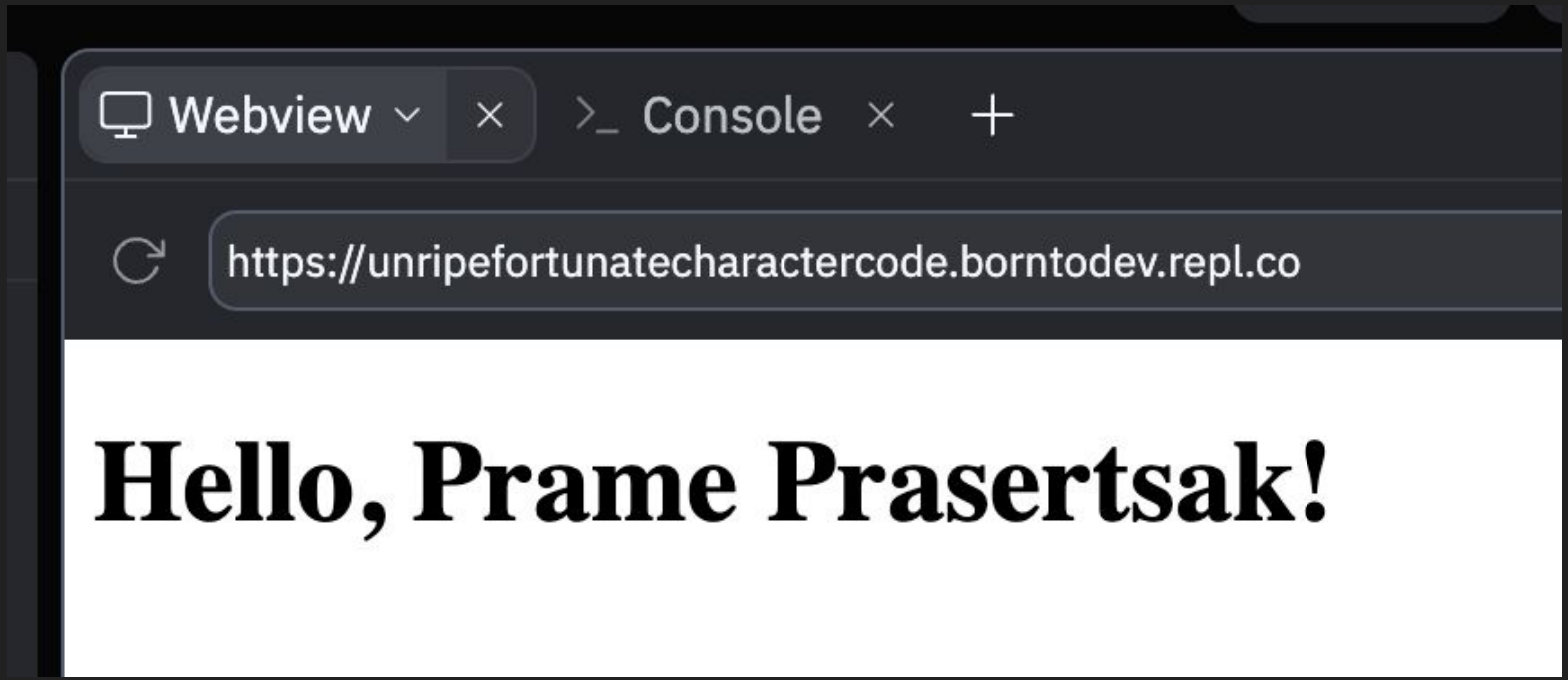
ลองเพิ่ม โค้ดนี้ ลงไป



ลองเพิ่ม โค้ดนี้ ลงไป



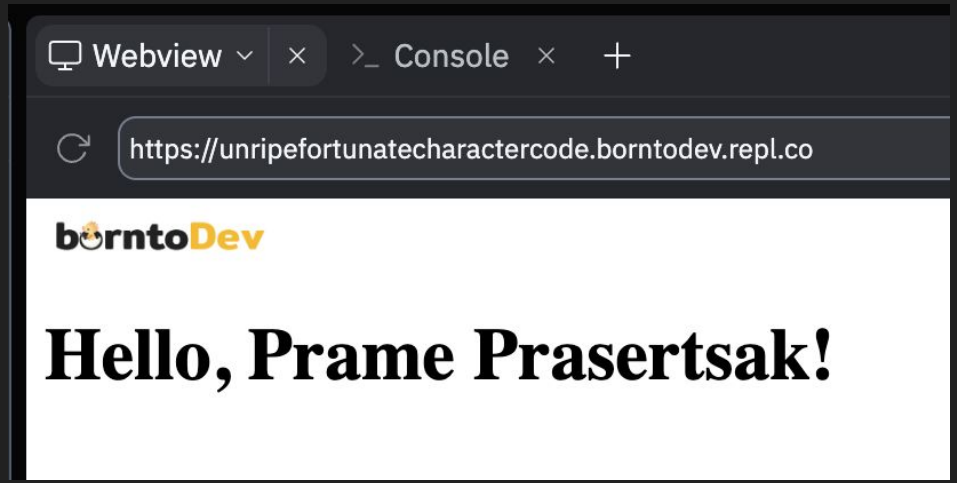
ลองเพิ่ม โค้ดนี้ ลงไป

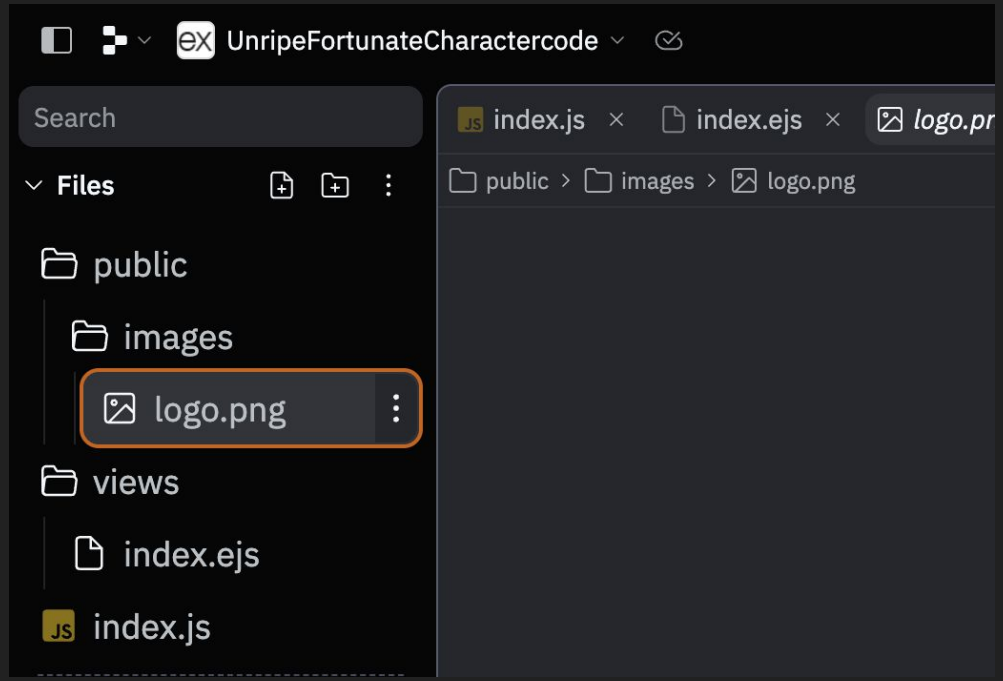


หากเราต้องการส่งข้อมูลภายใน Object

ไป 2 ตัว แล้วแสดง ชื่อ + นามสกุลหละ ?

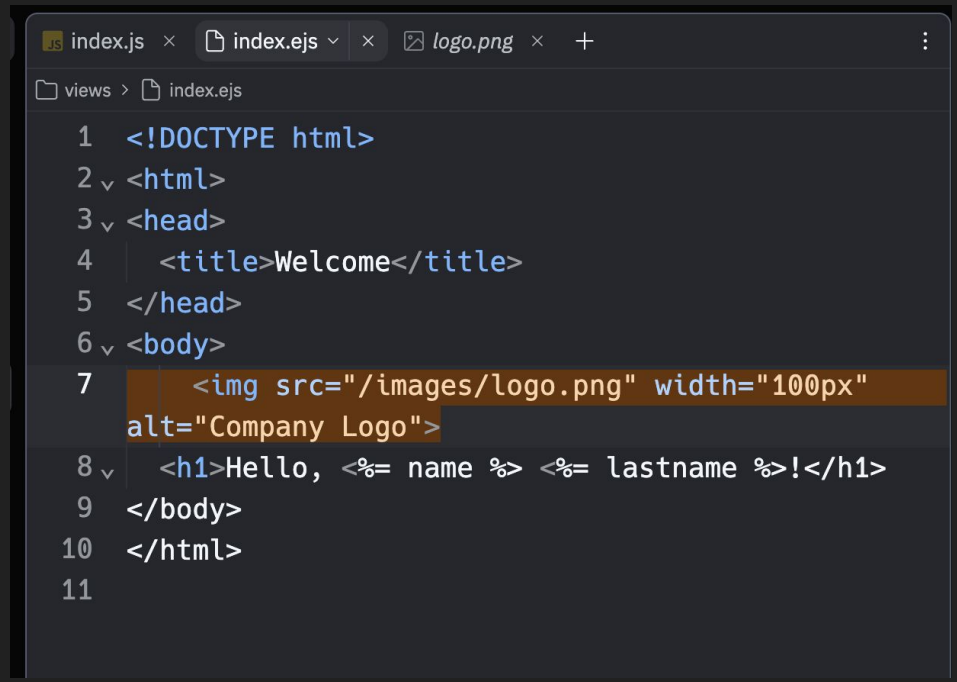
การนำ Static Content มาแสดง ?



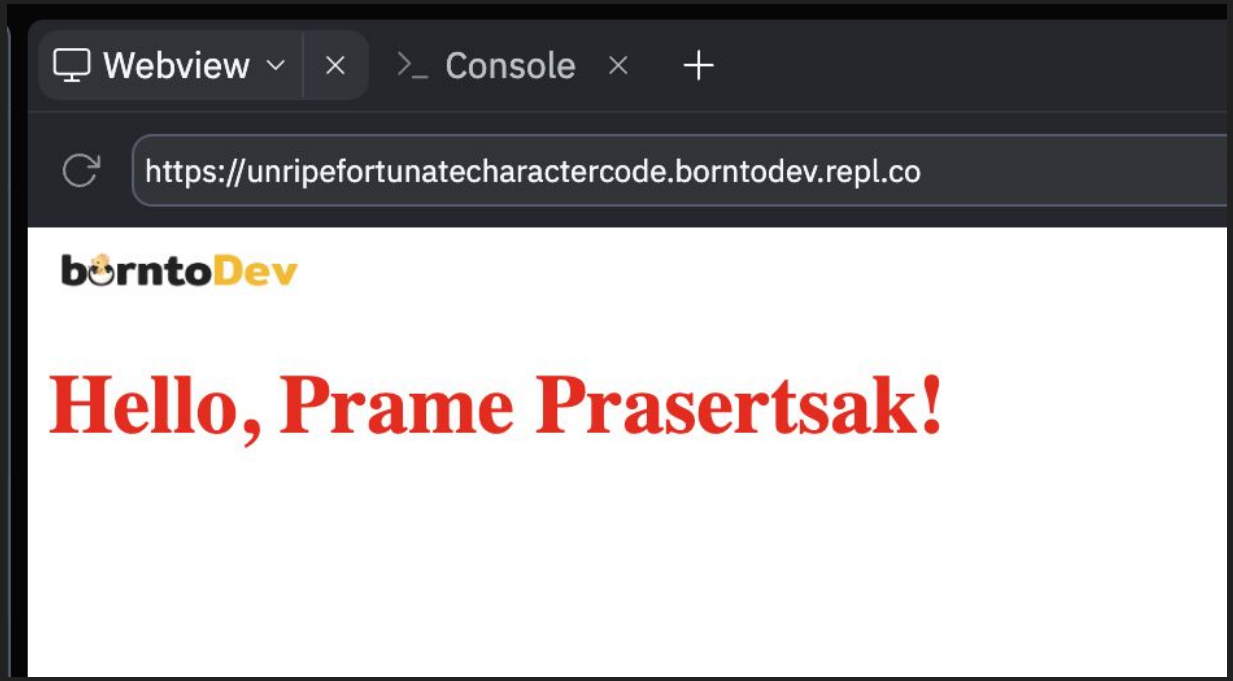


สร้าง Folder : public > images และ อัพรูปลงไป

เพิ่มการใช้งาน static ลงไปใน index.js



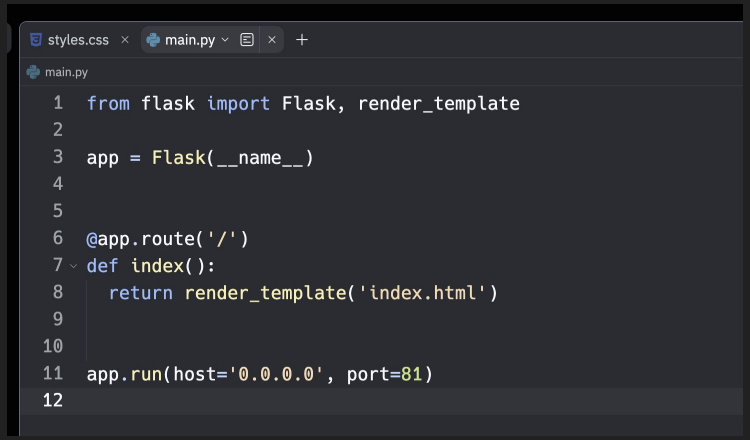
นำรูปภาพไปใส่ให้แสดงผลได้เลย



ลองทำกับ css ดูสิ !

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

ส่งข้อมูลมาเป็น Array ก็ได้

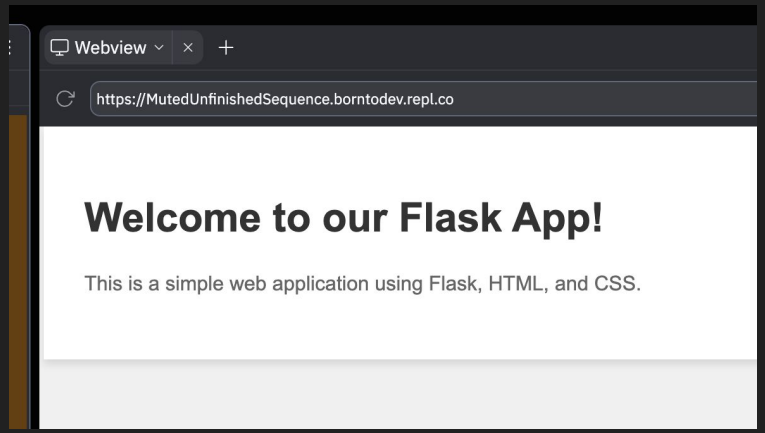


เสร็จแล้วกด Run ได้เลย

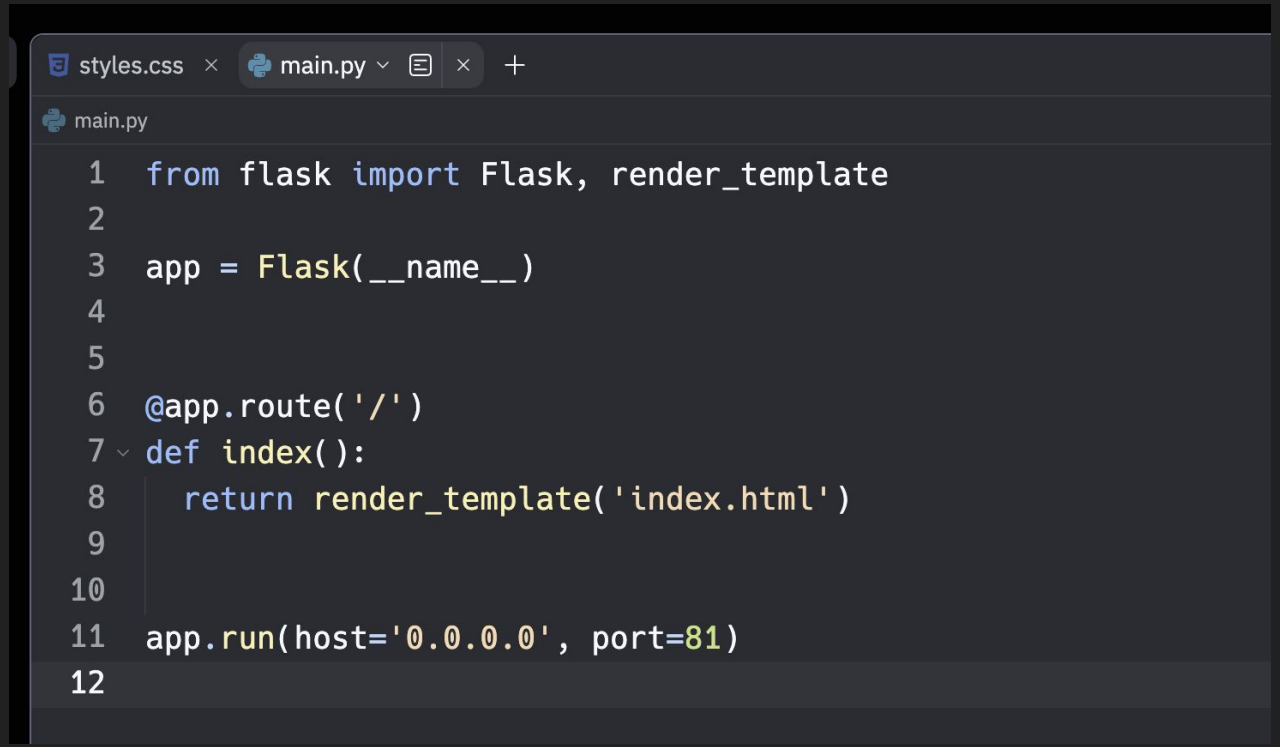
styles.css × main.py × +

main.py

```python
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)

```

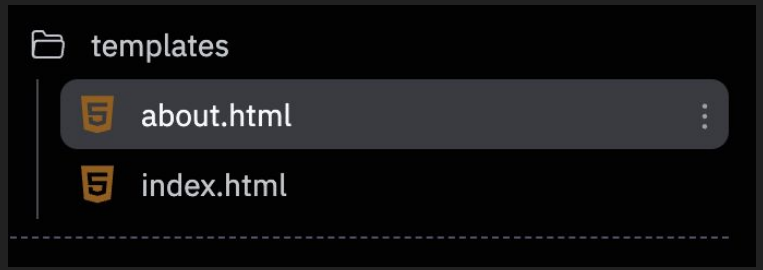


เสร็จแล้วกด Run ได้เลย



มาดูโค้ดตรงส่วนนี้กันดีกว่า

ถ้าเราต้องการ “เพิ่มหน้าใหม่” ลงไปหละ ?



styles.css about.html

templates > about.html

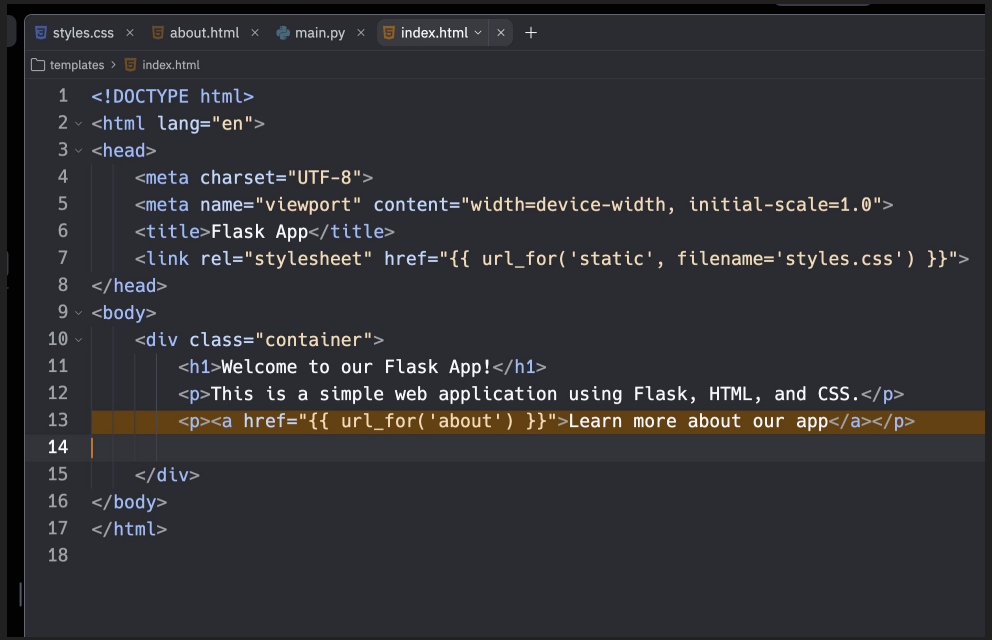
```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>About - Flask App</title>
7      <link rel="stylesheet" href="{{ url_for('static', filename='styles.css') }}">
8  </head>
9  <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>
```

สร้างไฟล์ about.html ลงไปใน templates

```python
from flask import Flask, render_template

app = Flask(__name__)


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

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


app.run(host='0.0.0.0', port=81)

```

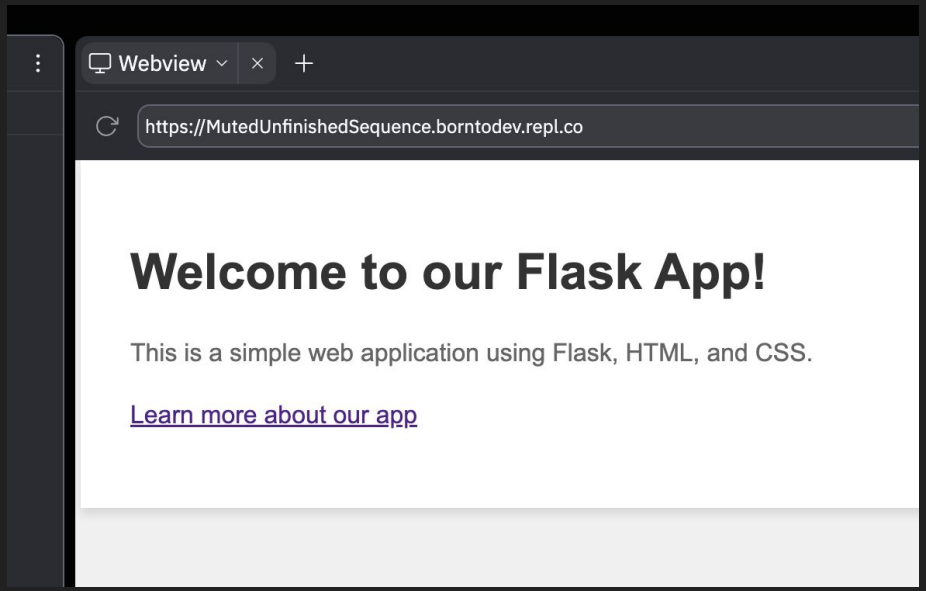
เพิ่ม @app.route ลงไปในส่วนของ about

ใส่ก่อน app.run เท่านั้นนะ !

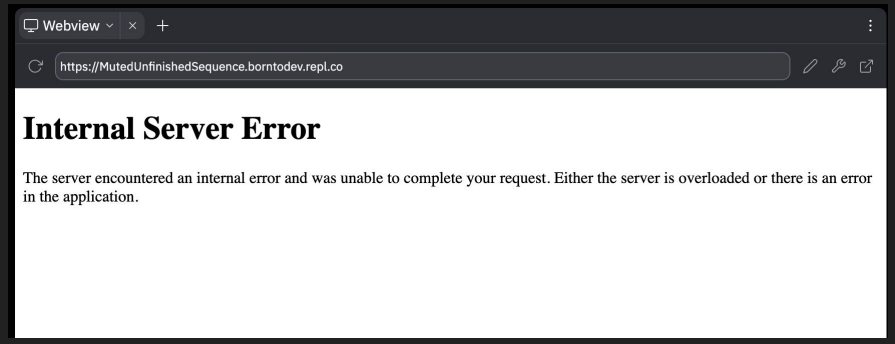


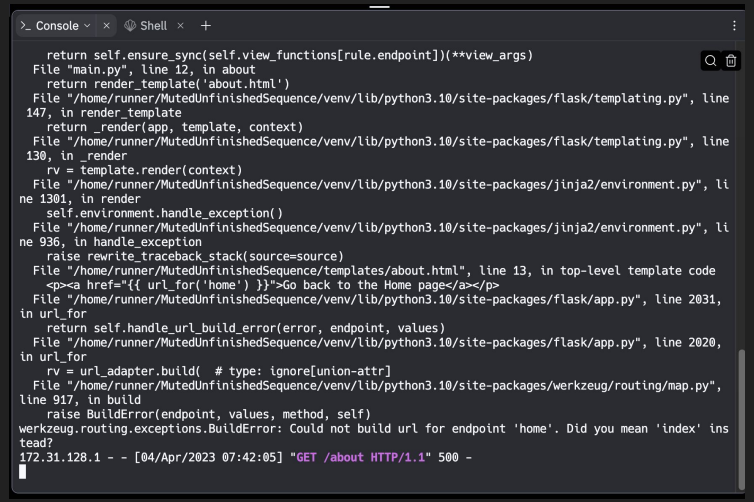
สุดท้ายเพิ่มลิงก์ลงไปใน index.html

แล้วลอง run ได้เลย !

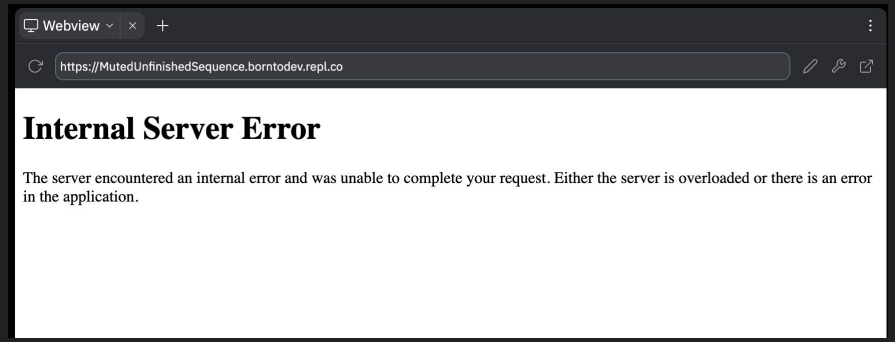


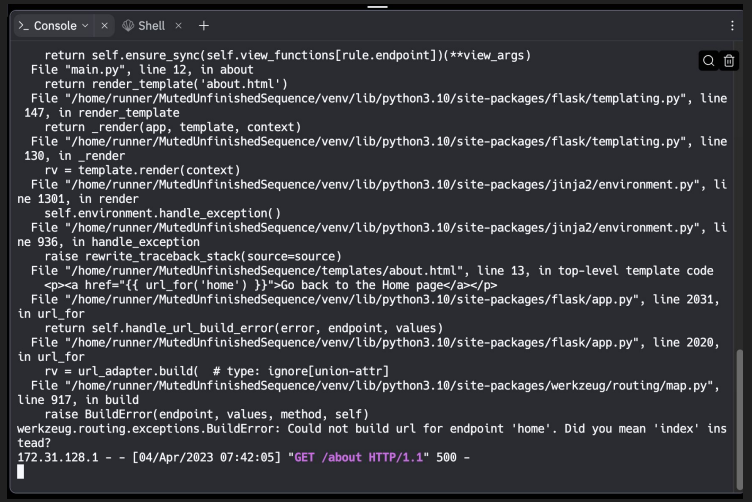
โอเค หน้าเว็บขึ้นมา ถือว่าผ่านไป 50% แล้ว



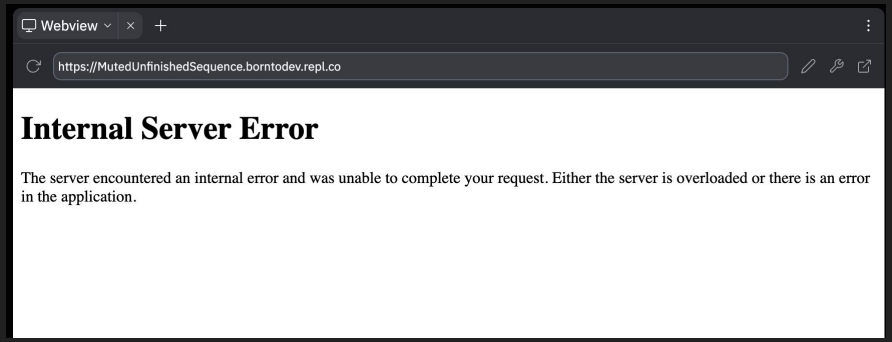


เมื่อคลิกไปที่ลิงก์จะพบว่ามัน .. Error



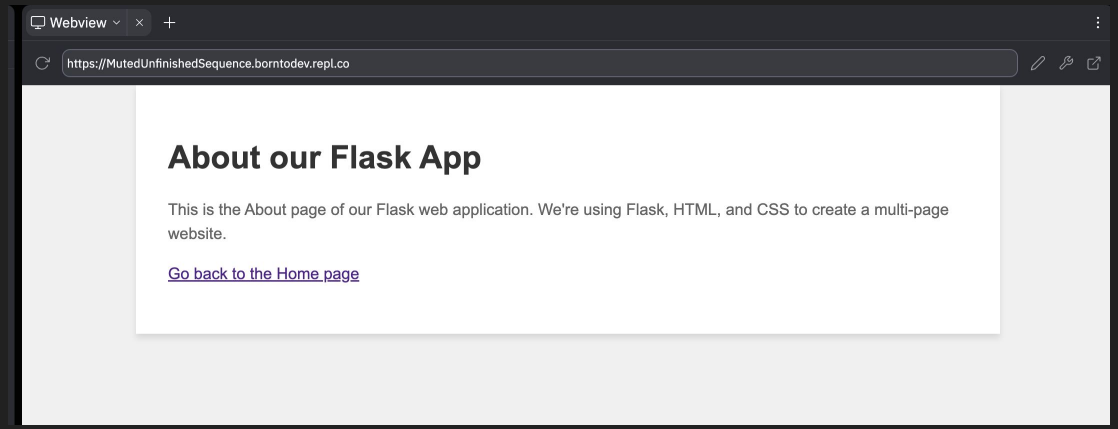


Quiz : มาลองเดาดูกันว่าเกิดจากอะไร ?




```
    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 -
```

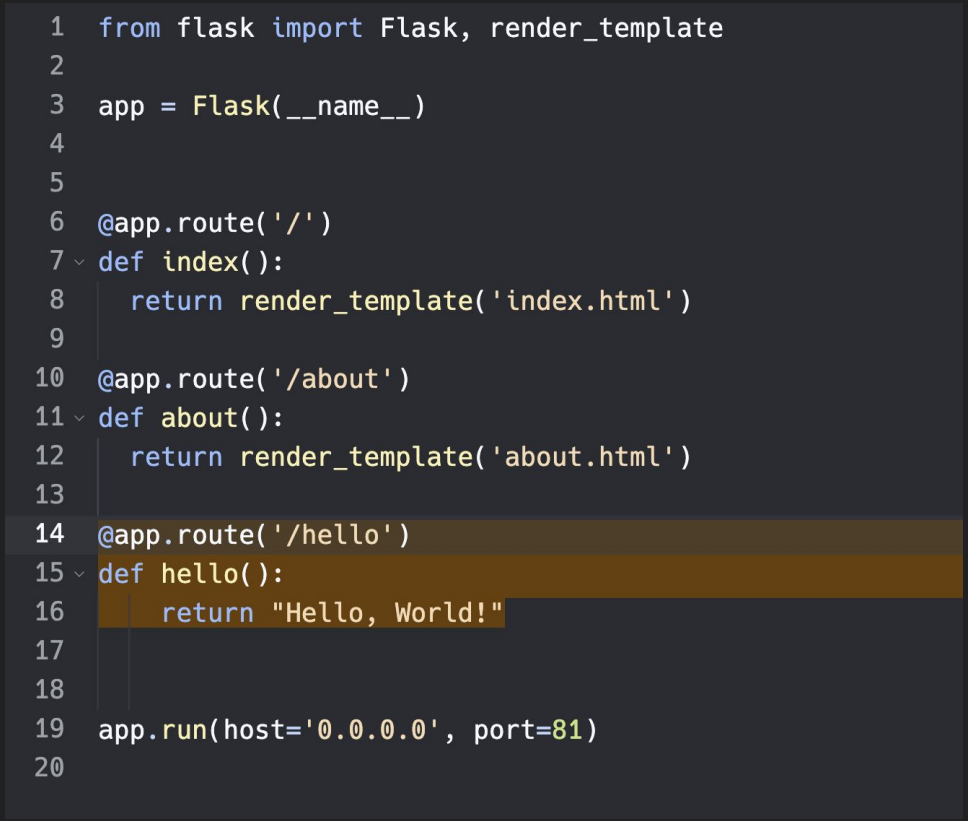
Quiz : มาลองเดาดูกันว่าเกิดจากอะไร ?



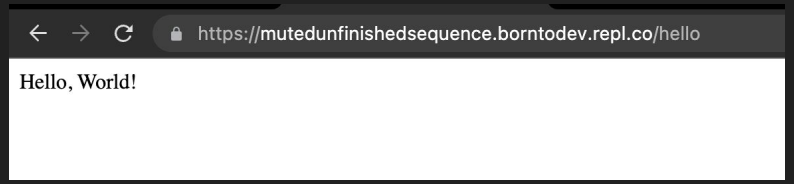
เมื่อแก้ไขเรียบร้อย หน้าเว็บก็จะออกมาปกตินั่นเอง

คำถามที่อยากให้ลองเล่น

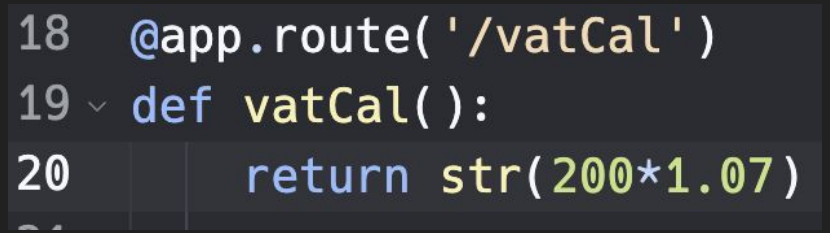
“พอเราใช้ Template แล้ว อยากจะให้ return มาตรง ๆ เหมือนเดิม ยังทำได้อยู่ไหม ?”


from flask import Flask, render_template
app = Flask(__name__)
@app.route('/')
def index():
return render_template('index.html')
@app.route('/about')
def about():
return render_template('about.html')
@app.route('/hello')
def hello():
return "Hello, World!"
app.run(host='0.0.0.0', port=81)

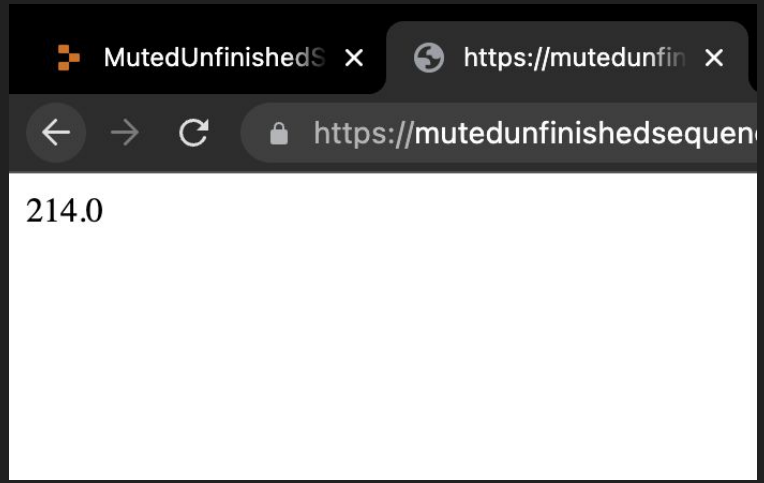
```python
from flask import Flask, render_template

app = Flask(__name__)


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

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

@app.route('/hello')
def hello():
    return "Hello, World!"


app.run(host='0.0.0.0', port=81)

```



คำถามต่อมาที่อยากให้ครูน่คริส
“พอ Flask มันใช้ Python งั้นเราก็เขียน
Code ส่วน Python ปกติลงไปได้ใช่ไหม ?”


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

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



เราควรเอาการประมวลผลนั้น ๆ

ไว้ที่ back / front ?

เราควรเอาการประมวลผลนั้น ๆ ไว้ที่ back / front ?

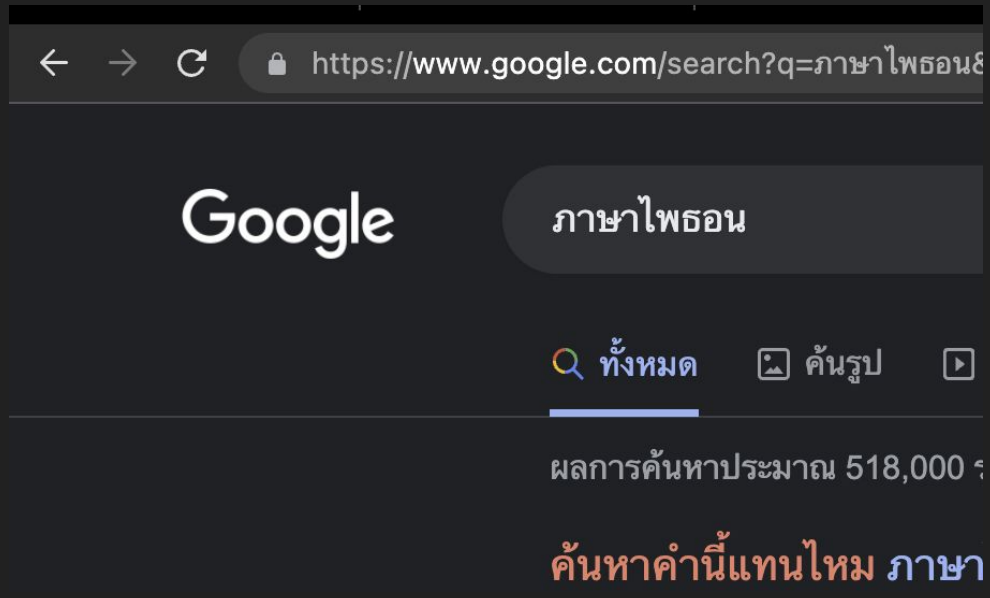
FE :
การตรวจสอบ Input, การตอบสนองกับหน้าตาแบบ real-time (responsive), การคำนวณประมวลผลทั่วไปที่ไม่จำเป็นต้องเก็บไว้ใน BE

BE :
การประมวลผลที่อาจจะต้องใช้การคำนวณที่ซับซ้อน สูตรเฉพาะ หรือ Algorithm ที่ไปจัดการแปลง เชื่อมโยงข้อมูลจำนวนมาก, การประมวลผลข้อมูลที่มีความ Sensitive หรือ เป็นความลับ เช่น การยืนยันตัวตัว การเข้าสู่ระบบ

หลายครั้งเราจำเป็ฯที่จะต้องทำทั้งฝั่ง
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>

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

```python
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

styles.css | about.html | greeting.html | index.html

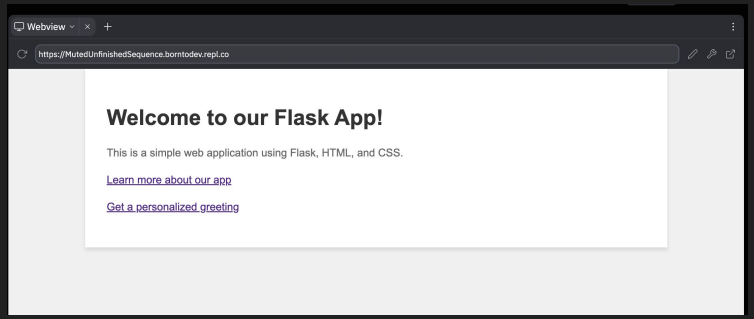
templates > greeting.html

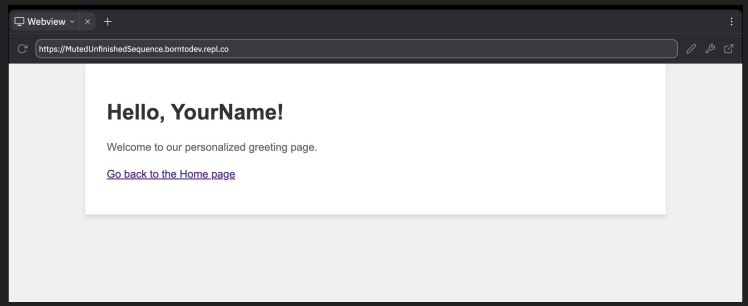
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Greeting - Flask App</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='styles.css') }}">
</head>
<body>
    <div class="container">
        <h1>Hello, {{ name }}!</h1>
        <p>Welcome to our personalized greeting page.</p>
        <p><a href="{{ url_for('index') }}">Go back to the Home page</a></p>
    </div>
</body>
</html>
```

ต่อมา ทำการสร้างหน้า greeting.html

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

สุดท้าย อัพเดตข้อมูลในหน้า 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,
    last_name=last_name)
29
```

ดำเนินการเพิ่ม app.route ใหม่ลงไป

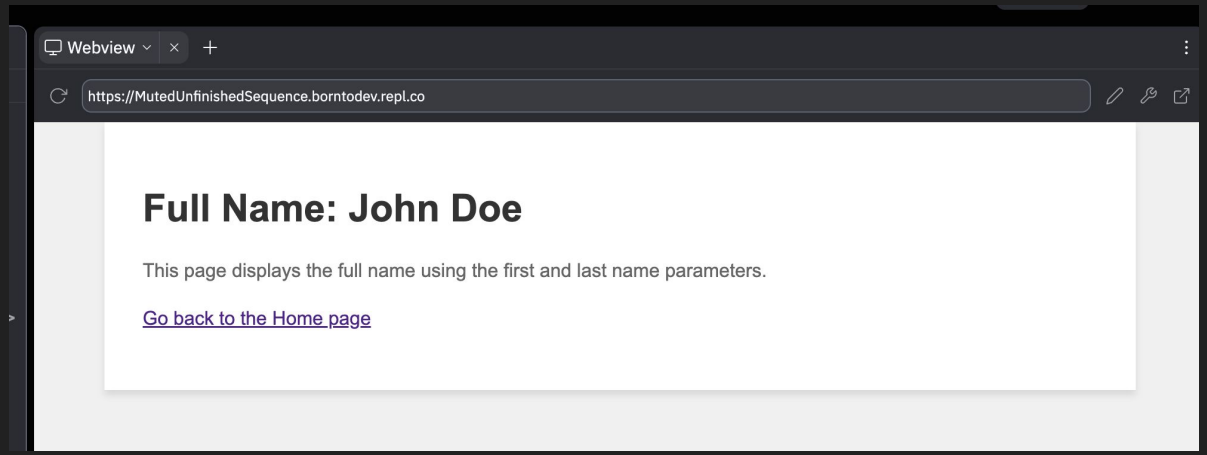
styles.css × about.html × greeting.html × index.html × full_name.html × +

templates › full_name.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Full Name - Flask App</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='styles.css') }}">
</head>
   <body>
       <div class="container">
           <h1>Full Name: {{ first_name }} {{ last_name }}</h1>
           <p>This page displays the full name using the first and last name
parameters.</p>
           <p><a href="{{ url_for('index') }}">Go back to the Home page</a></p>
       </div>
   </body>
   </html>
```

สร้างไฟล์ full_name.html ใน templates

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



มาทดสอบกันได้เลย

พามา Workshop
(กับเวลาที่เหลืออยู่อันน้อยนิด)

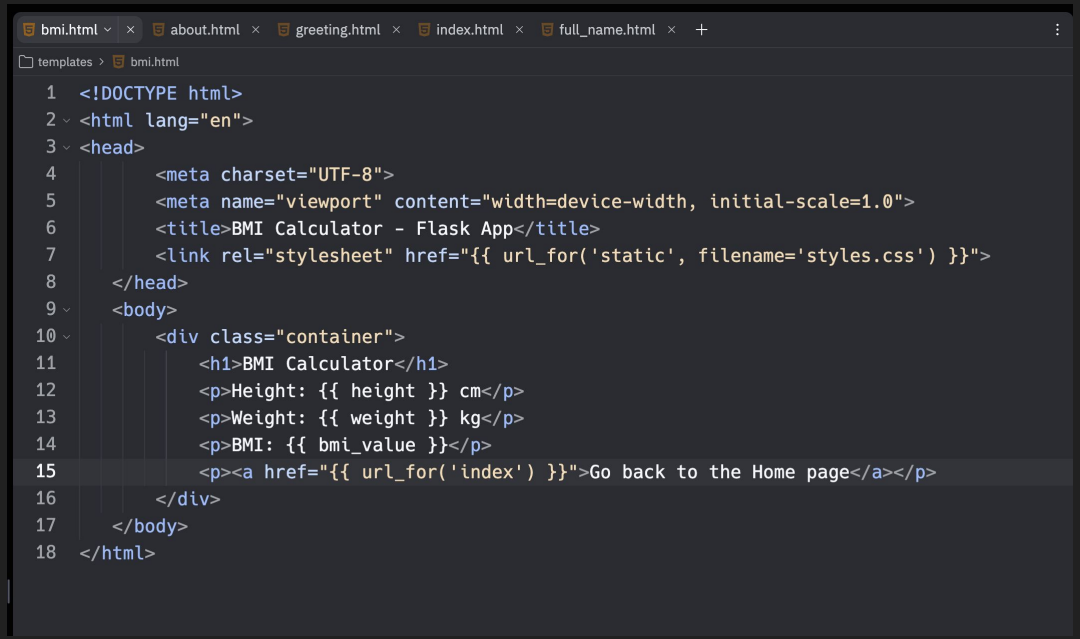
“ให้ลองสร้างหน้าใหม่ชื่อ bmiCalculate
โดยรับตัวเลขมาทั้งหมด 2 ตัว แล้วทำการ
รับตัวเลขน้ำหนัก และ ส่วนสูง แล้วส่งตัวเลขที่
คำนวณได้
ออกมาบนหน้าเว็บของเรา”

พามา Workshop
(กับเวลาที่เหลืออยู่อันน้อยนิด)

"ให้ลองสร้างหน้าใหม่ชื่อ bmiCalculate โดยรับตัวเลขมาทั้งหมด 2 ตัว แล้วทำการรับตัวเลขน้ำหนัก และ ส่วนสูง แล้วส่งตัวเลขที่คำนวณได้ออกมาบนหน้าเว็บของเรา"

เพิ่ม app.route การรับข้อมูล ลงไป

เพิ่ม app.route การรับข้อมูล ลงไป
สังเกตได้ว่าเราสามารถระบุประเภทข้อมูลที่จะโยนเข้าไปได้ !



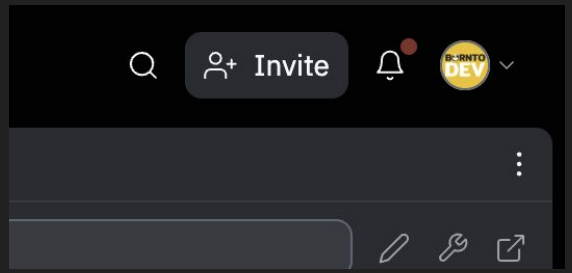
สร้างไฟล์ bmi.html เพื่อรอรับข้อมูลเข้ามา

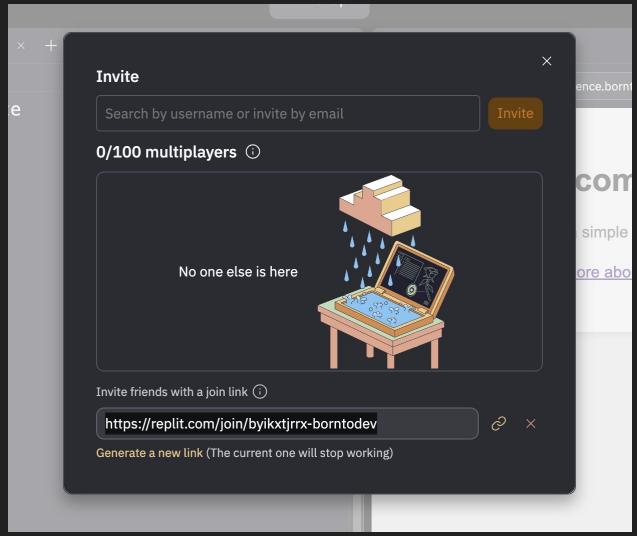
```
17
18      <p><a href="{{ url_for('bmi', height=170, weight=70) }}">Calculate example BMI</a></p>
19
```

สุดท้ายเพิ่มลิงก์ลงไปใน index.html

ตอบใน “ตอบคำถาม”

การส่งการบ้านด้วย Repl.it

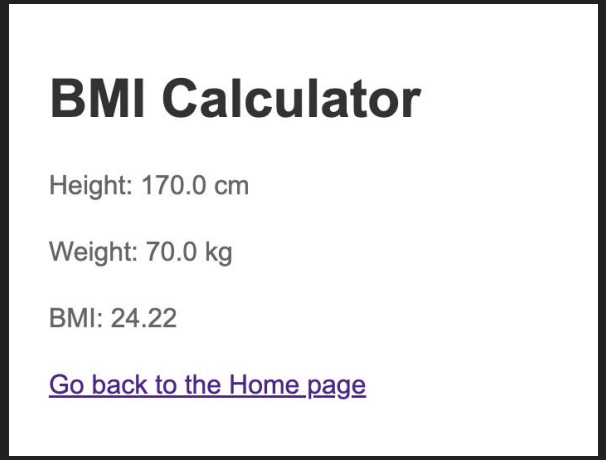




ให้สร้างลิงก์จากปุ่ม Invite และ กด Generate Link แล้ว
ดำเนินการคัดลอกมาใส่ใน ตอบคำถาม

ตอบใน “ตอบคำถาม”

การส่งการบ้านด้วย Repl.it



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

เพิ่มข้อความผลลัพธ์ของ BMI ออกมาบนหน้าจอ