

INFS3202/7202 – Web Information Systems

Lecture Week 12: Transitioning to Other Web Frameworks

Dr Aneesha Bakharia (Senior Lecturer, EECS)
Email: a.bakharia1@uq.edu.au

Contents

01 Course Updates

02 Recap of CodeIgniter MVC Concepts

03 Django

04 React

05 Next.JS

06 Other frameworks

Course Updates

Issue/Feedback	Change
SECaTS Open Today	<ul style="list-style-type: none">Please provide feedback on INFS3202/7202
Project Assessment Item	<ul style="list-style-type: none">Due Friday Week 12 at 3pm1 week remaining

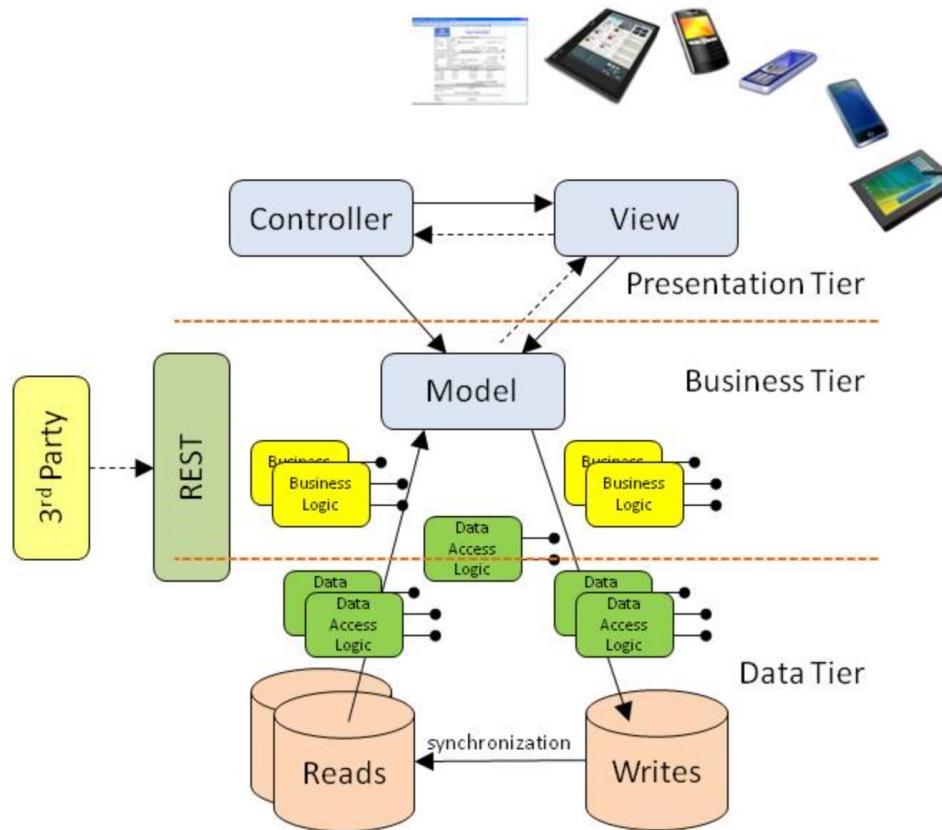
Learning Objectives

From Course Profile

1. Apply system architecture principles to design and deploy Web Information Systems (WIS) solutions.
2. Evaluate and articulate the scope, complexity, and key considerations in the design and implementation of Web Information Systems.
3. Design and program Web Information Systems (WIS) with server-side functionalities.
4. Develop responsive Web-based, database-driven applications using efficient and effective technologies.
5. Evaluate and justify the suitability of Web Information Systems solutions in various contexts, considering factors such as user needs and technical constraints.
6. Judge in which situations WIS solutions are more or less appropriate.
7. Critically analyze current issues and emerging trends in Web Information Systems development, and predict potential impacts on future practices and technologies.

Recap of CodeIgniter MVC Concepts

Model View Controller (MVC)



The MVC pattern is a specialized form of 3-layer architecture, making it highly relevant to modern web development practices

Image from https://en.wikiversity.org/wiki/Three-Tier_Architecture#/media/File:MVC3tierArchitecture.jpg

Convention vs Configuration

Convention vs. Configuration is a design philosophy that:

- impacts how software frameworks are structured (i.e. where files are located)
- how they expect developers to interact with them
- It's a trade-off between the ease of following predefined paths and files (convention) or the flexibility to define everything explicitly (configuration).
- Important advice:
 - Web frameworks (CodeIgniter, Django, Ruby on Rails, Flask, Next.js) come with a predefined folder structure
 - Learn what the structure is and where files go
 - This will improve your productivity!

How is the CI 4.4 Welcome Message displayed?

http://localhost:8080

The screenshot shows a code editor interface with two panes. The left pane is the Explorer view, showing the project structure under 'CODEIGNITERTEST'. It includes 'app', 'Controllers' (with 'HomeController.php' selected), 'Database', 'Filters', 'Helpers', 'Language', 'Libraries', 'Models', 'ThirdParty', 'Views' (with 'welcome_message.php' selected), and 'errors'. The right pane is the code editor showing 'Home.php' in the 'app > Controllers' directory. The code is:

```
<?php  
namespace App\Controllers;  
class Home extends BaseController  
{  
    public function index(): string  
    {  
        return view('welcome_message');  
    }  
}
```

A white arrow points from the 'welcome_message.php' file in the Explorer to the 'view('welcome_message')' line in the code editor. Another white arrow points from the 'welcome_message.php' file in the Explorer to the 'welcome_message.php' file in the code editor.

- / path runs the index() method
- the view function specifies the .PHP file to run
- The .PHP file can contain HTML and embedded PHP
- Welcome_message.php contains the “Welcome to CI 4.4 message”
- Flow of connections between files is very important! Understanding this is very important.

CodeIgniter – Mapping a Model to a DB Table

- **Model Configuration:** The model class provides configuration options to facilitate the smooth functioning of its methods.
 - **\$table:** Specifies the database table that this model primarily works with.
 - **\$primaryKey:** Specifies the name of the column that uniquely identifies the records in this table.
 - **\$useAutoIncrement:** Specifies if the table uses an auto-increment feature for \$primaryKey.
 - **\$returnType:** This setting allows you to define the type of data that is returned.
 - **\$allowedFields:** This array should be updated with the field names that can be set during save(), insert(), or update() methods. A

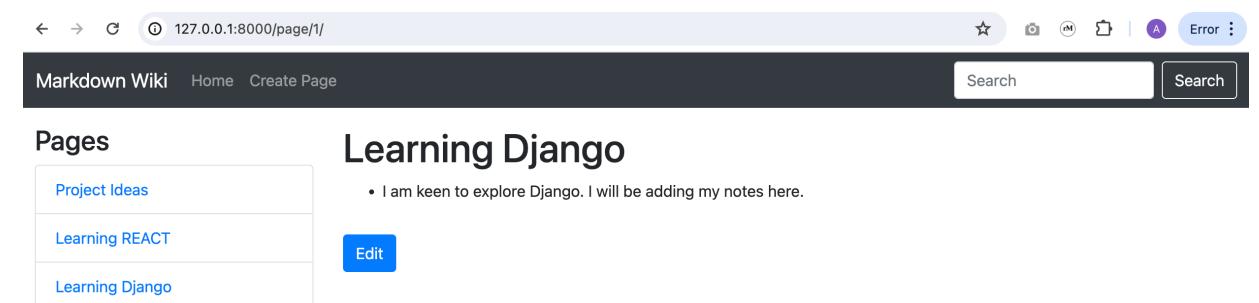
```
cidemo > app > Models > Customers.php
1  <?php
2
3  namespace App\Models;
4
5  use CodeIgniter\Model;
6
7  class Customers extends Model
8  {
9      protected $table      = 'customers';
10     protected $primaryKey   = 'id';
11     protected $useAutoIncrement = true;
12     protected $returnType    = 'array';
13     protected $useSoftDeletes = false;
14     protected $protectFields = true;
15     protected $allowedFields = [];
16
17 }
```

How to transition to other web frameworks?

- Web fundamentals are the same.
 - HTML
 - Javascript
 - Forms (Post vs Get)
- MVC implementations are very similar
 - Eg Flask, Django, FastAPI
 - But there are differences
 - And different conventions and folder structures
 - And more powerful Models (including Object Relational Modeling) than the models in CodeIgniter
 - And different templating libraries (i.e. alternatives to embedded PHP in HTML)
 - And different ways to implement Routing
- More recently there are newer ideas
 - React (client-side)
 - Next.JS (React that runs on Server with new ideas for layouts)

Comparing Frameworks

- We will look at Django, React and Next.js
- We will cover:
 - Installation
 - Project Structure
 - Routing and Controllers
 - How to make a base template?
 - Models
- We will build a small WIKI app



Django (Server-side Python Framework)

Django

The screenshot shows the official Django website at djangoproject.com. The header features the Django logo and the tagline "The web framework for perfectionists with deadlines." Navigation links include OVERVIEW, DOWNLOAD, DOCUMENTATION, NEWS, COMMUNITY, CODE, ISSUES, ABOUT, a DONATE button, and a search icon. The main content area has a teal background with white text. It starts with the headline "Django makes it easier to build better web apps more quickly and with less code." followed by a large green button labeled "Get started with Django". Below this, there's a section titled "Meet Django" with a paragraph about the framework's design and history, and a "Download latest release: 5.0.6" button. The footer contains sections for "Support Django!", "Latest news", and a "download apk mod games" link.

Meet Django

Django is a high-level Python web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of web development, so you can focus on writing your app without needing to reinvent the wheel. It's free and open source.



Ridiculously fast.

Django was designed to help developers take applications from concept to completion as quickly as possible.

[Download latest release: 5.0.6](#)

[DJANGO DOCUMENTATION >](#)

Support Django!



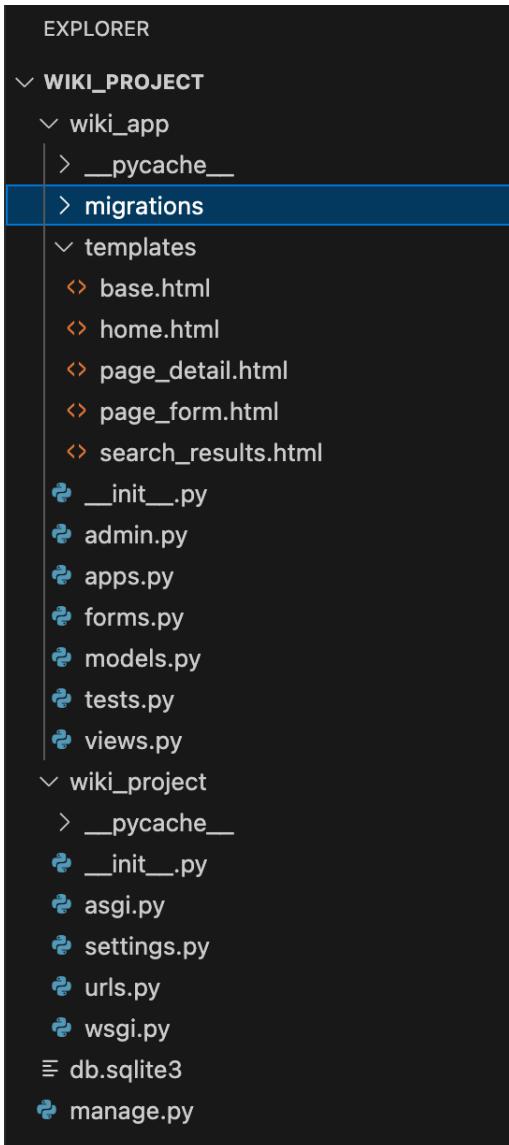
download apk mod games donated to the Django Software Foundation to support Django development. Donate today!

Latest news

Creating a Django Project

- **Create a virtual environment**
 - `python3 -m venv venv`
 - `source venv/bin/activate`
- **Install Django and a Markdown library**
 - `pip install django`
 - `pip install markdown`
- **Create a Django project and app**
 - `django-admin startproject wiki_project`
 - `python manage.py startapp wiki_app`

Django Project Structure



The main directories are:

- **wiki_app**: Contains the application-specific code, including models, views, forms, and templates.
- **wiki_project**: Contains the project-level configuration files and settings.
 - **settings.py**: The main Django project settings file.
 - **urls.py**: The URL configuration file for the project.

Django Project Structure – Settings.py

```
wiki_project > 📄 settings.py
1     """
2     Django settings for wiki_project project.
3
4     Generated by 'django-admin startproject' using Django 5.0.6.
5
6     For more information on this file, see
7     https://docs.djangoproject.com/en/5.0/topics/settings/
8
9     For the full list of settings and their values, see
10    https://docs.djangoproject.com/en/5.0/ref/settings/
11    """
12
13    from pathlib import Path
14
15    # Build paths inside the project like this: BASE_DIR / 'subdir'.
16    BASE_DIR = Path(__file__).resolve().parent.parent
17
18
19    # Quick-start development settings - unsuitable for production
20    # See https://docs.djangoproject.com/en/5.0/howto/deployment/checklist/
21
22    # SECURITY WARNING: keep the secret key used in production secret!
23    SECRET_KEY = 'django-insecure-fw7i$00a-kxh+5&)69utciul=4mij50jc5_jk5+g=2=$f2&yj4'
24
25    # SECURITY WARNING: don't run with debug turned on in production!
26    DEBUG = True
27
28    ALLOWED_HOSTS = []
29
30
31    # Application definition
32
33    INSTALLED_APPS = [
34        'django.contrib.admin',
35        'django.contrib.auth',
36        'django.contrib.contenttypes',
37        'django.contrib.sessions',
38        'django.contrib.messages',
39        'django.contrib.staticfiles',
40        'wiki_app', # Add this line
41    ]
```

Django Project Structure – urls.py

```
wiki_project > 📄 urls.py
1  """
2  URL configuration for wiki_project project.
3
4  The `urlpatterns` list routes URLs to views. For more information please see:
5      https://docs.djangoproject.com/en/5.0/topics/http/urls/
6  Examples:
7  Function views
8      1. Add an import: from my_app import views
9         2. Add a URL to urlpatterns: path('', views.home, name='home')
10 Class-based views
11     1. Add an import: from other_app.views import Home
12     2. Add a URL to urlpatterns: path('', Home.as_view(), name='home')
13 Including another URLconf
14     1. Import the include() function: from django.urls import include, path
15     2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))
16 """
17
18 from django.contrib import admin
19 from django.urls import path
20 from wiki_app.views import home, page_detail, search, page_form
21
22 urlpatterns = [
23     path('admin/', admin.site.urls),
24     path('', home, name='home'),
25     path('page/<int:pk>/', page_detail, name='page_detail'),
26     path('search/', search, name='search'),
27     path('page/add/', page_form, name='add_page'),
28     path('page/<int:pk>/edit/', page_form, name='edit_page')
29 ]
```

Creating a Model for the WIKI

```
wiki_app > 🐍 models.py
1   from django.db import models
2
3   class Page(models.Model):
4       title = models.CharField(max_length=100)
5       content = models.TextField()
6       created_at = models.DateTimeField(auto_now_add=True)
7
8       def __str__(self):
9           return self.title
```

```
[(venv) uqabakh1@uqabakh1-9893 wiki_project % python manage.py makemigrations
Migrations for 'wiki_app':
  - Create model Page
[(venv) uqabakh1@uqabakh1-9893 wiki_project % python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions, wiki_app
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying auth.0012_alter_user_first_name_max_length... OK
  Applying sessions.0001_initial... OK
  Applying wiki_app.0001_initial... OK
```

In Django we don't create the migration first.

We create a model using simple syntax. Then we generate the migration. Django can convert a model into the migration file.

This is much easier than CodeIgniter 😊

The Autogenerated Migration

The screenshot shows a code editor interface with a dark theme. On the left is the Explorer sidebar, which lists the project structure of 'WIKI_PROJECT'. The 'migrations' folder contains '0001_initial.py', which is currently selected and highlighted with a blue background. The main workspace displays the contents of '0001_initial.py'. The code is as follows:

```
models.py  0001_initial.py ×  views.py  page_form.html  base.html  home.html  page_detail.html  
wiki_app > migrations > 0001_initial.py  
1 # Generated by Django 5.0.6 on 2024-05-11 10:53  
2  
3 from django.db import migrations, models  
4  
5  
6 class Migration(migrations.Migration):  
7     initial = True  
8  
9     dependencies = [  
10         ]  
11  
12     operations = [  
13         migrations.CreateModel(  
14             name='Page',  
15             fields=[  
16                 ('id', models.BigAutoField(auto_created=True, primary_key=True, serialize=False, verbose_name='ID')),  
17                 ('title', models.CharField(max_length=100)),  
18                 ('content', models.TextField()),  
19                 ('created_at', models.DateTimeField(auto_now_add=True)),  
20             ],  
21         ),  
22     ],  
23  
24 ]
```

Base Templates

```
wiki_app > templates > base.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>Markdown Wiki</title>
7      <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
8  </head>
9  <body>
10     <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
11         <a class="navbar-brand" href="{% url 'home' %}">Markdown Wiki</a>
12         <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarNav"
13             aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">
14             <span class="navbar-toggler-icon"></span>
15         </button>
16         <div class="collapse navbar-collapse" id="navbarNav">
17             <ul class="navbar-nav mr-auto">
18                 <li class="nav-item">
19                     <a class="nav-link" href="{% url 'home' %}">Home</a>
20                 </li>
21                 <li class="nav-item">
22                     <a class="nav-link" href="{% url 'add_page' %}">Create Page</a>
23                 </li>
24             </ul>
25             <form class="form-inline my-2 my-lg-0" action="{% url 'search' %}" method="get">
26                 <input class="form-control mr-sm-2" type="search" name="q" placeholder="Search" aria-label="Search">
27                 <button class="btn btn-outline-light my-2 my-sm-0" type="submit">Search</button>
28             </form>
29         </div>
30     </nav>
31
32     <div class="container-fluid mt-3">
33         <div class="row">
34             <div class="col-md-3">
35                 <h3>Pages</h3>
36                 <ul class="list-group">
37                     {% for page in pages %}
38                         <li class="list-group-item">
39                             <a href="{% url 'page_detail' page.pk %}">{{ page.title }}</a>
40                         </li>
41                     {% endfor %}
42                 </ul>
43             </div>
44             <div class="col-md-9">
45                 {% block content %}{% endblock %}
46             </div>
47         </div>
48     </div>
49 
```

- We can embed Python within HTML.
- This uses Jinja Templating
- We can also define blocks
- Blocks are like the CodeIgniter sections.

```
wiki_app > templates > home.html
1  {% extends 'base.html' %}
2
3  {% block content %}
4      <h1>Welcome to Markdown Wiki</h1>
5
6      <a href="{% url 'add_page' %}" class="btn btn-primary mt-3">Create New Page</a>
7  {% endblock %}
```

Views

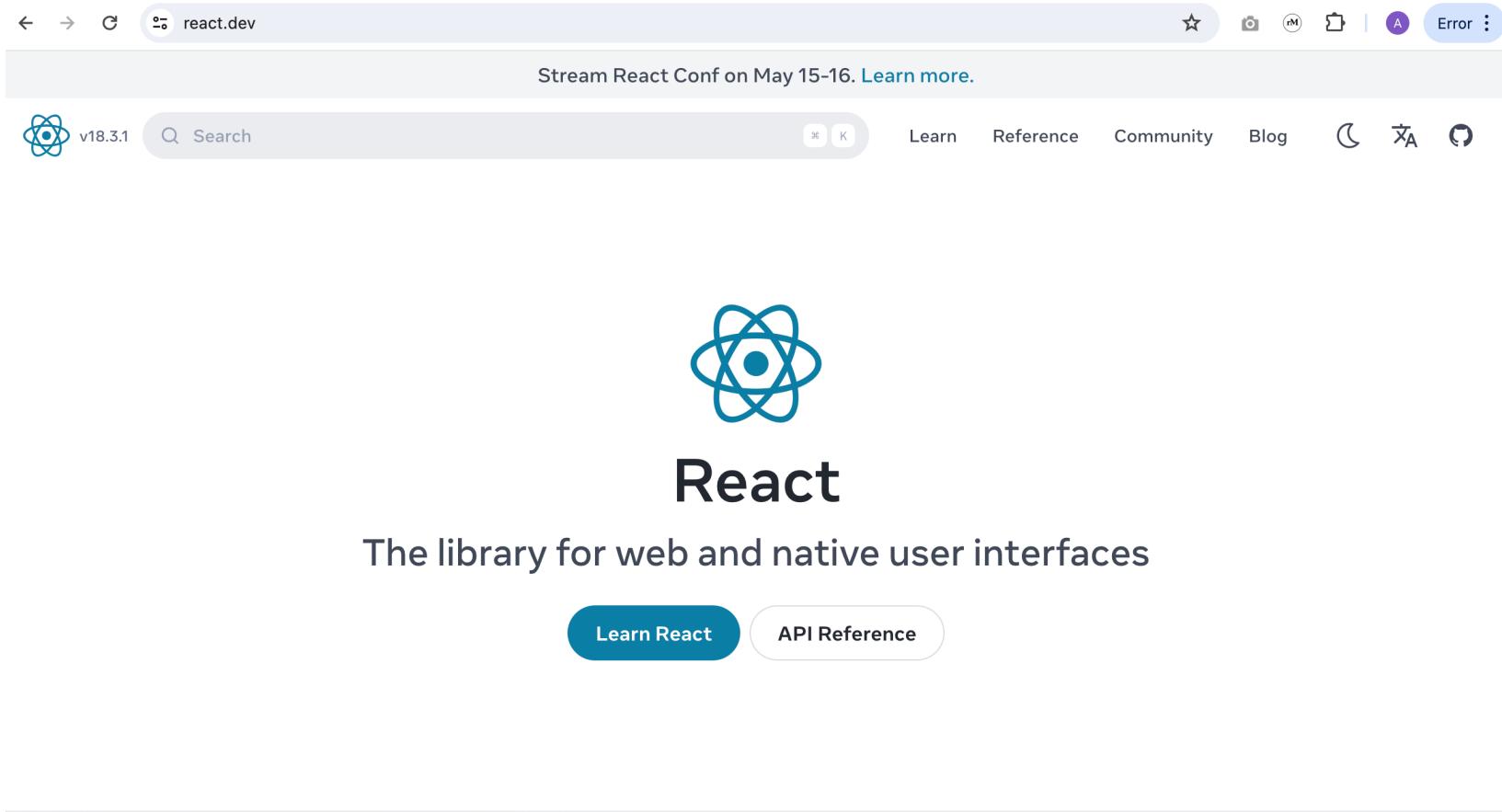
```
wiki_app > 🌐 views.py
 1  from django.shortcuts import render, redirect, get_object_or_404
 2  from .models import Page
 3  from .forms import PageForm
 4  import markdown
 5
 6  def get_pages():
 7      return Page.objects.order_by('-created_at')
 8
 9  def home(request):
10      pages = get_pages()
11      return render(request, 'home.html', {'pages': pages})
12
13  def page_detail(request, pk):
14      page = get_object_or_404(Page, pk=pk)
15      page.content = markdown.markdown(page.content)
16      pages = get_pages()
17      return render(request, 'page_detail.html', {'page': page, 'pages': pages})
18
19  def search(request):
20      query = request.GET.get('q')
21      pages = Page.objects.filter(title__icontains=query)
22      all_pages = get_pages()
23      return render(request, 'search_results.html', {'pages': pages, 'query': query, 'all_pages': all_pages})
24
25  def page_form(request, pk=None):
26      if pk:
27          page = get_object_or_404(Page, pk=pk)
28      else:
29          page = None
30
31      if request.method == 'POST':
32          form = PageForm(request.POST, instance=page)
33          if form.is_valid():
34              page = form.save()
35              return redirect('page_detail', pk=page.pk)
36      else:
37          form = PageForm(instance=page)
38
39      pages = get_pages()
40      return render(request, 'page_form.html', {'form': form, 'pages': pages})
```

- The views.py file contains methods that can use models and render templates
- This is essentially the equivalent of the CodeIgniter Controllers
- Django Models are much more powerful. Django has a full Object Relational Mapper (ORM)
- Data is passed to the template

React

(Client-side Javascript Framework)

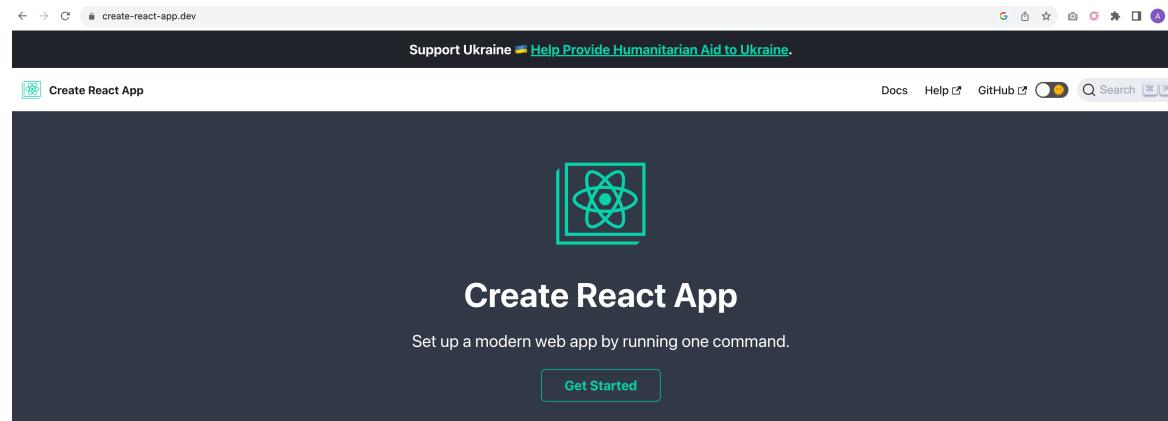
React



A screenshot of the React homepage on a web browser. The URL in the address bar is `react.dev`. The page features a navigation bar with links for `v18.3.1`, `Search`, `Learn`, `Reference`, `Community`, `Blog`, and other icons for dark mode, accessibility, and error reporting. A banner at the top says "Stream React Conf on May 15-16. [Learn more.](#)". The main content area has the React logo (atom icon) and the word "React" in large letters, followed by the subtitle "The library for web and native user interfaces". Below this are two buttons: "Learn React" (in a teal box) and "API Reference".

Using create-react-app

- **Create a new react client side application:**
 - `npx create-react-app simple-app --use-npm`
- **What does create-react-app do?**
 - A command-line tool developed by Meta (Facebook) that sets up a new React project with a recommended file structure and configuration.
 - It simplifies the process of starting a React application by automatically setting up a development environment, including the necessary dependencies and build scripts, without requiring manual configuration.



Less to Learn

You don't need to learn and configure many build tools. Instant reloads help you focus on development. When it's time to deploy, your bundles are optimized automatically.

Only One Dependency

Your app only needs one build dependency. We test Create React App to make sure that all of its underlying pieces work together seamlessly – no complicated version mismatches.

No Lock-In

Under the hood, we use webpack, Babel, ESLint, and other amazing projects to power your app. If you ever want an advanced configuration, you can "eject" from Create React App and edit their config files directly.

Creating a React Project

- Create a new react client side application:

- npx create-react-app simple-app --use-npm
- cd simple-app
- npm start

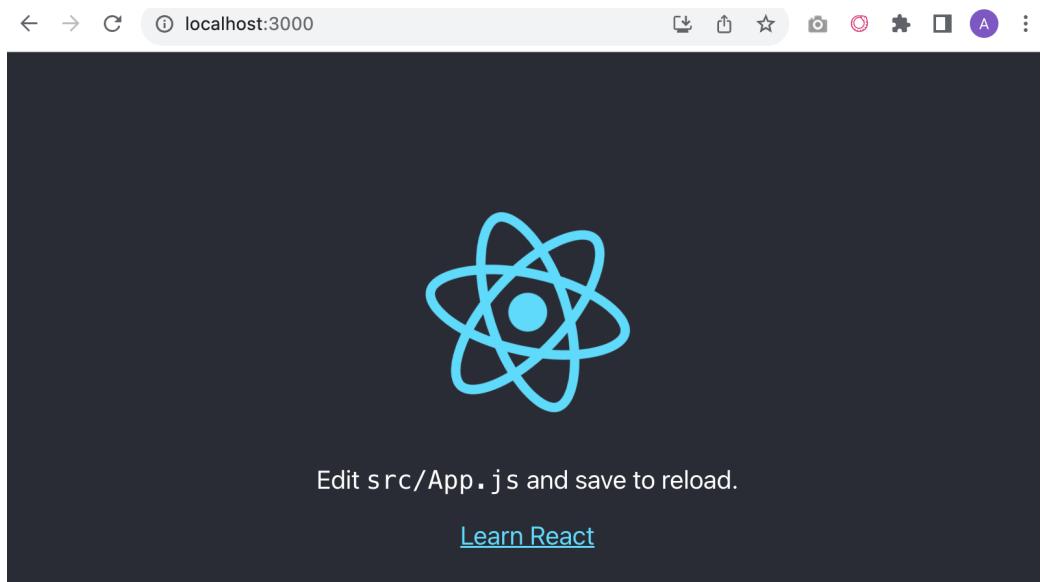
```
Compiled successfully!
```

```
You can now view simple-app in the browser.
```

```
Local:          http://localhost:3000
On Your Network: http://10.0.0.113:3000
```

```
Note that the development build is not optimized.
To create a production build, use npm run build.
```

```
webpack compiled successfully
```



Explore the created folders

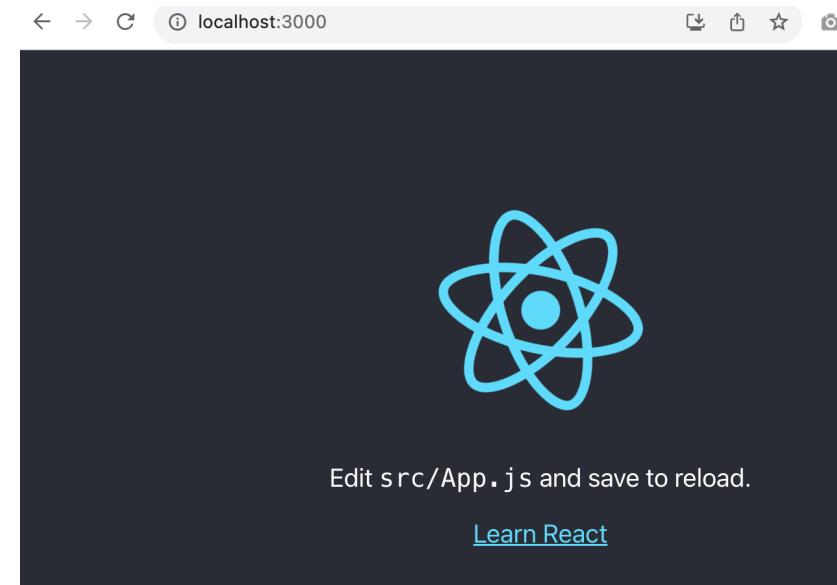
EXPLORER ...

SIMPLE-APP

- > node_modules
- > public
- src
- # App.css
- JS App.js**
- JS App.test.js
- # index.css
- JS index.js
- logo.svg
- JS reportWebVitals.js
- JS setupTests.js
- .gitignore
- { package-lock.json
- { package.json
- README.md

JS App.js

```
src > JS App.js > ...
1 import logo from './logo.svg';
2 import './App.css';
3
4 function App() {
5   return (
6     <div className="App">
7       <header className="App-header">
8         <img src={logo} className="App-logo" alt="logo" />
9         <p>
10           Edit <code>src/App.js</code> and save to reload.
11         </p>
12         <a
13           className="App-link"
14           href="https://reactjs.org"
15           target="_blank"
16           rel="noopener noreferrer"
17         >
18           Learn React
19         </a>
20       </header>
21     </div>
22   );
23 }
24
25 export default App;
```



HelloWorld Component

```
src > components > JS HelloWorld.js > [e] default
1 import React from 'react';
2
3 function HelloWorld() {
4   return (
5     <h1>Hello, World!</h1>
6   );
7 }
8
9 export default HelloWorld;
```

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



localhost:3000

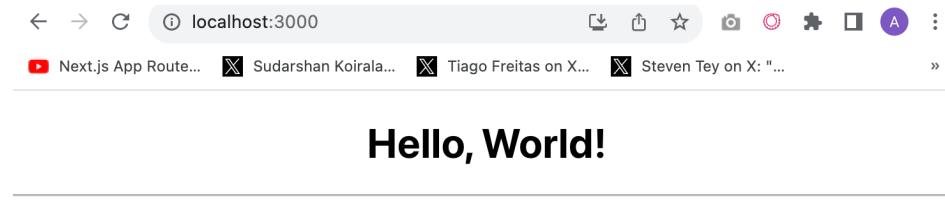


Hello, World!

Header Component

```
src > components > JS Header.js > [d] default
1   import React from 'react';
2
3   function Header(props) {
4     return (
5       <h1>{props.text}</h1>
6     );
7   }
8
9   export default Header;
```

```
src > JS App.js > ...
1   import HelloWorld from './components/HelloWorld';
2   import Header from './components/Header';
3   import './App.css';
4
5   function App() {
6     return (
7       <div className="App">
8         <HelloWorld />
9         <hr />
10        <Header text="Header Level 1" />
11      </div>
12    );
13  }
14
15  export default App;
16
```



ItemList Component

```
src > components > JS ItemList.js > ...
1  import React from 'react';
2
3  function ItemList() {
4      const items = ['Item 1', 'Item 2', 'Item 3'];
5
6      return (
7          <ul>
8              {items.map((item, index) => (
9                  <li key={index}>{item}</li>
10             ))}
11         </ul>
12     );
13 }
14
15 export default ItemList;
```

Hello, World!

Header Level 1

- Item 1
- Item 2
- Item 3

```
src > JS App.js > ...
1  import HelloWorld from './components/HelloWorld';
2  import Header from './components/Header';
3  import ItemList from './components/ItemList';
4  import './App.css';
5
6  function App() {
7      return (
8          <div className="App">
9              <HelloWorld />
10             <hr />
11             <Header text="Header Level 1" />
12             <hr />
13             <ItemList />
14         </div>
15     );
16 }
17
18 export default App;
```

Add Item with a Button

```
src > components > JS ItemListWithState.js > ...
1  import React, { useState } from 'react';
2
3  function ItemListWithState() {
4    const [items, setItems] = useState(['Item 1', 'Item 2', 'Item 3']);
5
6    const addItem = () => {
7      const newItem = `Item ${items.length + 1}`;
8      setItems([...items, newItem]);
9    };
10
11   return (
12     <div style={{ marginTop: '20px', display: 'grid', placeItems: 'center' }}>
13       <ul>
14         {items.map((item, index) => (
15           <li key={index}>{item}</li>
16         ))}
17       </ul>
18       <button onClick={addItem}>Add Item</button>
19     </div>
20   );
21 }
22
23 export default ItemListWithState;
24 |
```

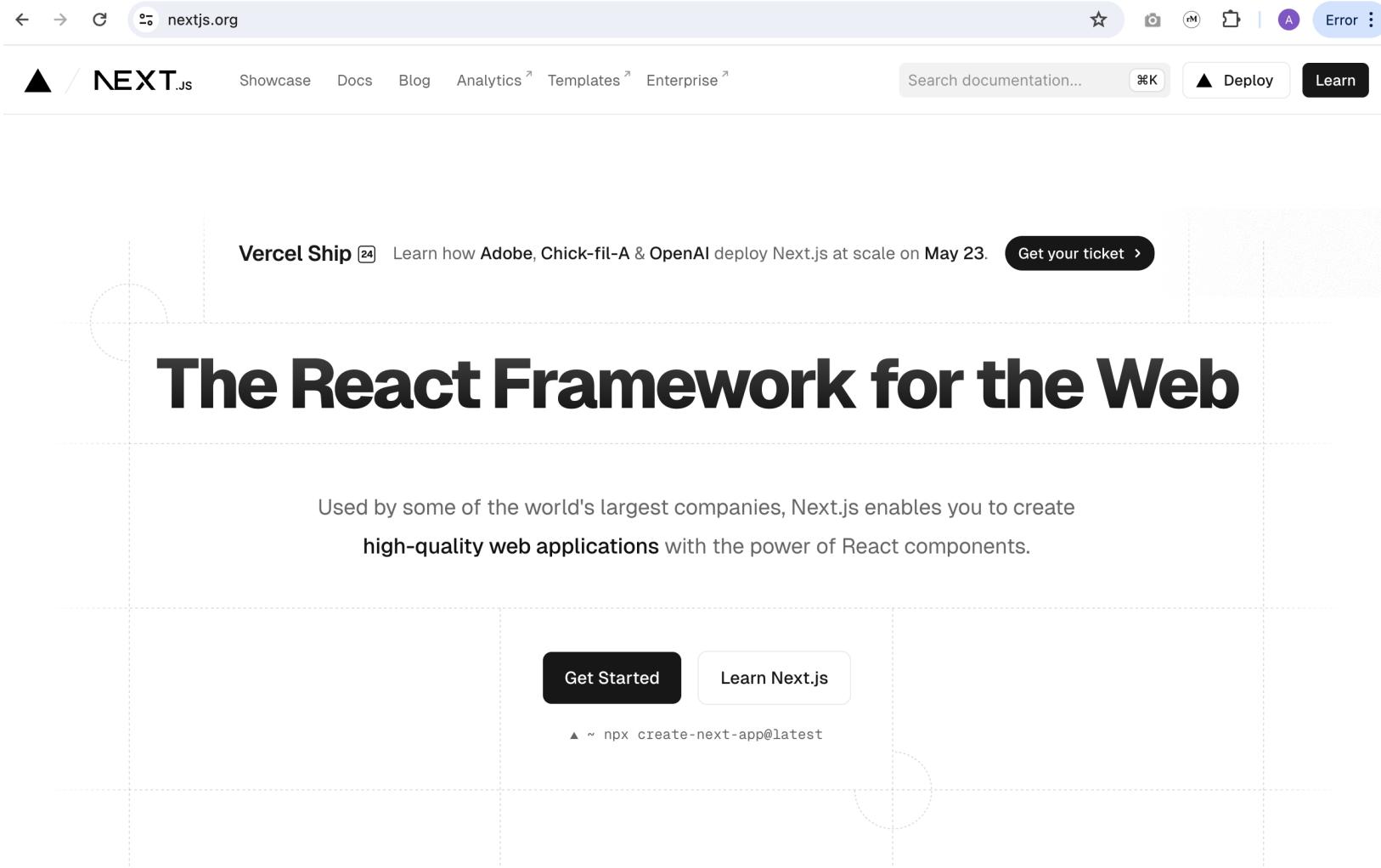
- Item 1
- Item 2
- Item 3
- Item 4

Add Item

- **What have we not had to program?**
- **State is a very powerful concept!**
- **When state is updated the component is re-rendered**

Next.JS

Next.js



The screenshot shows the official Next.js website at nextjs.org. The page features a large, bold title "The React Framework for the Web". Below the title, a subtitle reads: "Used by some of the world's largest companies, Next.js enables you to create high-quality web applications with the power of React components." At the bottom of the main content area, there are two buttons: "Get Started" (in a dark box) and "Learn Next.js" (in a light box). A command-line interface (CLI) command, "`▲ ~ npx create-next-app@latest`", is displayed below the buttons. The top navigation bar includes links for Showcase, Docs, Blog, Analytics, Templates, Enterprise, and a search bar. On the right side of the header, there are icons for star, camera, file, and error, along with a "Deploy" button.

nextjs.org

Showcase Docs Blog Analytics Templates Enterprise

Search documentation... `⌘K`

Deploy Learn

Vercel Ship 24 Learn how Adobe, Chick-fil-A & OpenAI deploy Next.js at scale on May 23. [Get your ticket >](#)

The React Framework for the Web

Used by some of the world's largest companies, Next.js enables you to create **high-quality web applications** with the power of React components.

Get Started Learn Next.js

`▲ ~ npx create-next-app@latest`

Using create-next-app

- **Install Node.js** <https://nodejs.org/en>
- **Create a new react client side application:**
 - `npx create-next-app@latest`

```
((venv) uqabakh1@uqabakh1-9893 nextjswiki % npx create-next-app@latest
Need to install the following packages:
create-next-app@14.2.3
[Ok to proceed? (y) y
[✓] What is your project named? ... nextjs-wiki
[✓] Would you like to use TypeScript? ... No / Yes
[✓] Would you like to use ESLint? ... No / Yes
[✓] Would you like to use Tailwind CSS? ... No / Yes
[✓] Would you like to use `src/` directory? ... No / Yes
[✓] Would you like to use App Router? (recommended) ... No / Yes
[✓] Would you like to customize the default import alias (@/*)? ... No / Yes
Creating a new Next.js app in /Users/uqabakh1/INFS3202/lectures/week12/week12_code/nextjswiki/nextjs-wiki.
```

Using npm.

```
Initializing project with template: app
```

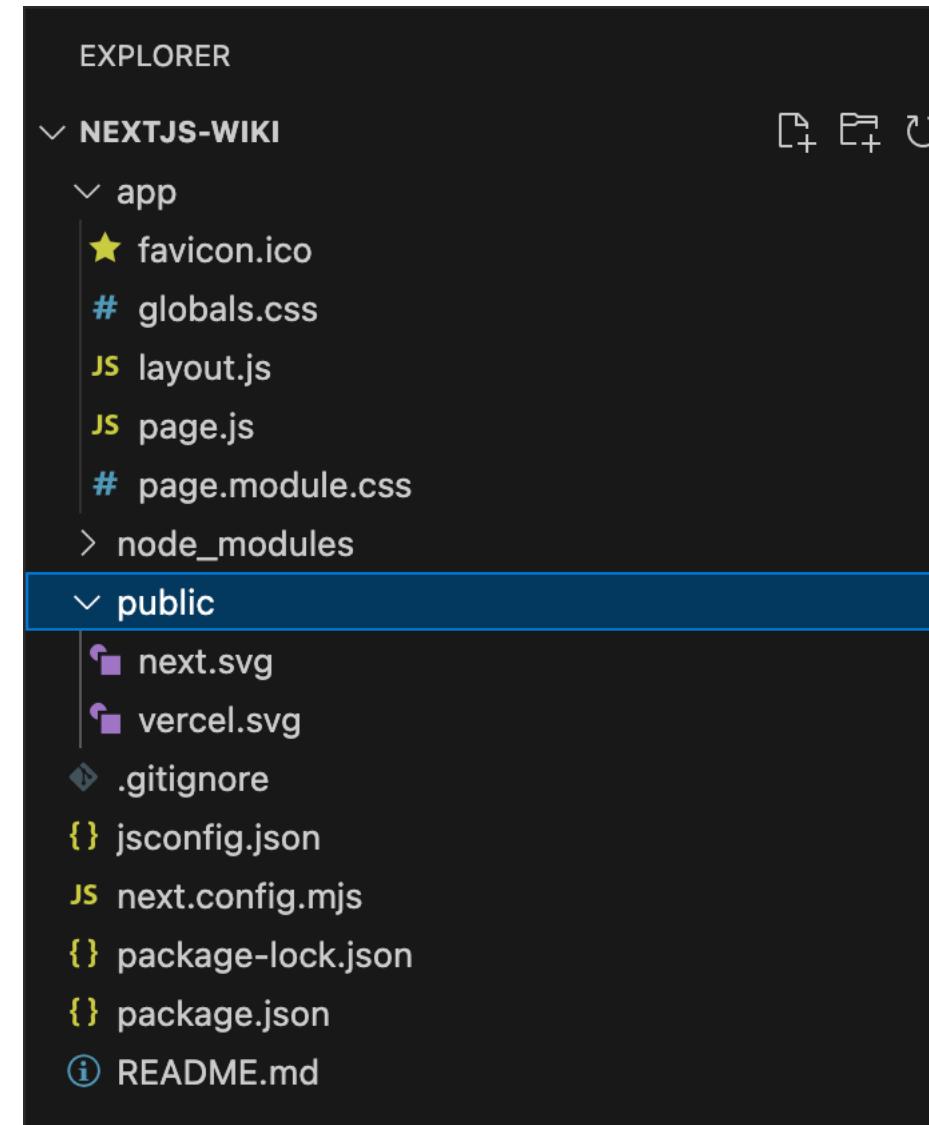
```
Installing dependencies:
```

```
- react
- react-dom
- next
```

```
added 21 packages, and audited 22 packages in 15s
```

```
3 packages are looking for funding
  run `npm fund` for details
```

```
found 0 vulnerabilities
Success! Created nextjs-wiki at /Users/uqabakh1/INFS3202/lectures/week12/week12_code/nextjswiki/nextjs-wiki
```



Install Dependencies

- We will use Prisma as our ORM and Bootstrap for CSS

- `npm install @prisma/client bootstrap`

- Initialise Prisma

- `npx prisma init`

```
(venv) uqabakh1@uqabakh1-9893 nextjs-wiki % npx prisma init

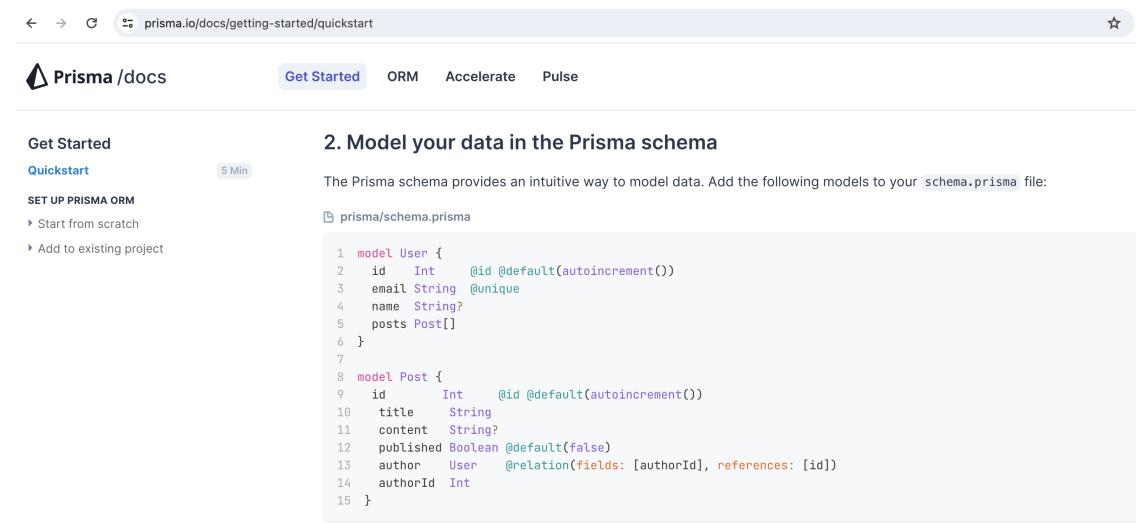
✓ Your Prisma schema was created at prisma/schema.prisma
  You can now open it in your favorite editor.

warn You already have a .gitignore file. Don't forget to add `*.env` in it to not commit any private information.

Next steps:
1. Set the DATABASE_URL in the .env file to point to your existing database. If your database has no tables yet, read https://pris.ly/d/getting-started
2. Set the provider of the datasource block in schema.prisma to match your database: postgresql, mysql, sqlite, sqlserver, mongodb or cockroachdb.
3. Run prisma db pull to turn your database schema into a Prisma schema.
4. Run prisma generate to generate the Prisma Client. You can then start querying your database.

More information in our documentation:
https://pris.ly/d/getting-started
```

Developing real-time features?
Prisma Pulse lets you respond instantly to database changes.
<https://pris.ly/cli/pulse>



The screenshot shows a browser window displaying the Prisma documentation at prisma.io/docs/getting-started/quickstart. The page title is "Prisma /docs". The main content area is titled "2. Model your data in the Prisma schema". It includes a brief description: "The Prisma schema provides an intuitive way to model data. Add the following models to your `schema.prisma` file:" followed by a code snippet for `schema.prisma`:

```
1 model User {
2   id Int @id @default(autoincrement())
3   email String @unique
4   name String?
5   posts Post[]
6 }
7
8 model Post {
9   id Int @id @default(autoincrement())
10  title String
11  content String?
12  published Boolean @default(false)
13  author User @relation(fields: [authorId], references: [id])
14  authorId Int
15 }
```

Models in the Prisma schema have two main purposes:

- Represent the tables in the underlying database
- Serve as foundation for the generated Prisma Client API

Create the schema (i.e. Model)

The screenshot shows a code editor interface with a dark theme. On the left, the Explorer panel displays a project structure for a Next.js application named 'NEXTJS-WIKI'. The 'prisma' folder contains a file named 'schema.prisma', which is currently selected and highlighted in the Explorer panel.

```
prisma > schema.prisma
1 // This is your Prisma schema file,
2 // learn more about it in the docs: https://pris.ly/d/prisma-schema
3
4 // Looking for ways to speed up your queries, or scale easily with your serverless or edge functions?
5 // Try Prisma Accelerate: https://pris.ly/cli/accelerate-init
6
7 datasource db {
8   provider = "sqlite"
9   url      = "file:./dev.db"
10 }
11
12 generator client {
13   provider = "prisma-client-js"
14 }
15
16 model Page {
17   id      Int    @id @default(autoincrement())
18   slug    String @unique
19   title   String
20   content String
21 }
22
23
```

Generate the Prisma Client

- Run this command to create the client

- npx prisma generate

Prisma Client is an auto-generated and type-safe query builder for Node.js

- **Auto-generated:** Prisma Client is automatically generated based on your Prisma schema file (schema.prisma). It provides an intuitive and type-safe API for querying and modifying your database.
- **Query builder:** Prisma Client provides a fluent and expressive query builder API. It allows you to construct complex database queries using a chain of methods, making your code more readable and maintainable.
- **CRUD operations:** Prisma Client supports all the common CRUD (Create, Read, Update, Delete) operations. It provides methods like create, findUnique, findMany, update, delete, etc., to perform these operations on your database entities.

```
[uqabakh1@uqabakh1-9893 nextjs-wiki % npx prisma generate
Environment variables loaded from .env
Prisma schema loaded from prisma/schema.prisma

✓ Generated Prisma Client (v5.13.0) to ./node_modules/@prisma/client in 49ms
Start using Prisma Client in Node.js (See: https://pris.ly/d/client)
```
import { PrismaClient } from '@prisma/client'
const prisma = new PrismaClient()
```
or start using Prisma Client at the edge (See: https://pris.ly/d/accelerate)
```
import { PrismaClient } from '@prisma/client/edge'
const prisma = new PrismaClient()
```

See other ways of importing Prisma Client: http://pris.ly/d/importing-client
```

Supercharge your Prisma Client with global database caching, scalable connection pooling and real-time database events.
Explore Prisma Accelerate: <https://pris.ly/cli/-accelerate>
Explore Prisma Pulse: <https://pris.ly/cli/-pulse>

Create the database table (similar to Migrations)

- Run this command to create the client

- npx prisma migrate dev --name init

```
[uqabakh1@uqabakh1-9893 nextjs-wiki % npx prisma migrate dev --name init
Environment variables loaded from .env
Prisma schema loaded from prisma/schema.prisma
Datasource "db": SQLite database "dev.db" at "file:./dev.db"
SQLite database dev.db created at file:./dev.db

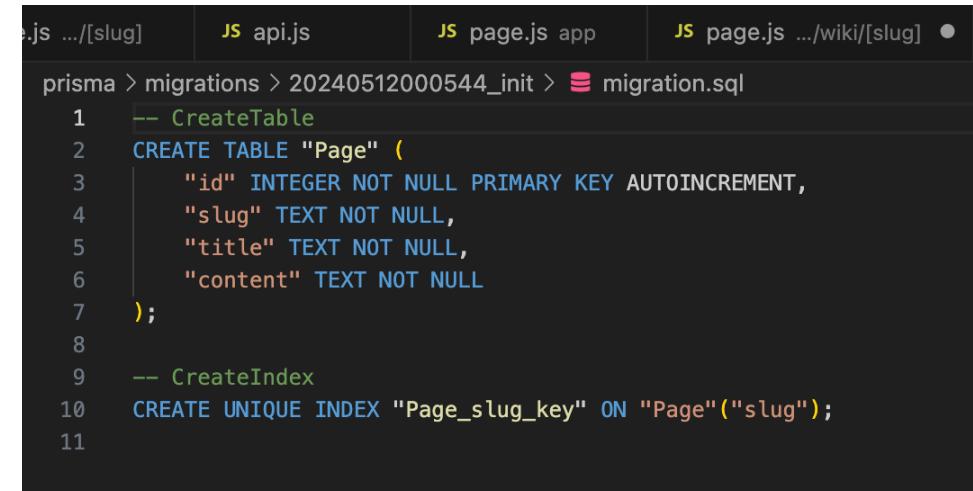
Applying migration `20240512000544_init`
```

The following migration(s) have been created and applied from new schema changes:

```
migrations/
└── 20240512000544_init/
    └── migration.sql
```

Your database is now in sync with your schema.

✓ Generated **Prisma Client** (v5.13.0) to ./node_modules/@prisma/client in 43ms

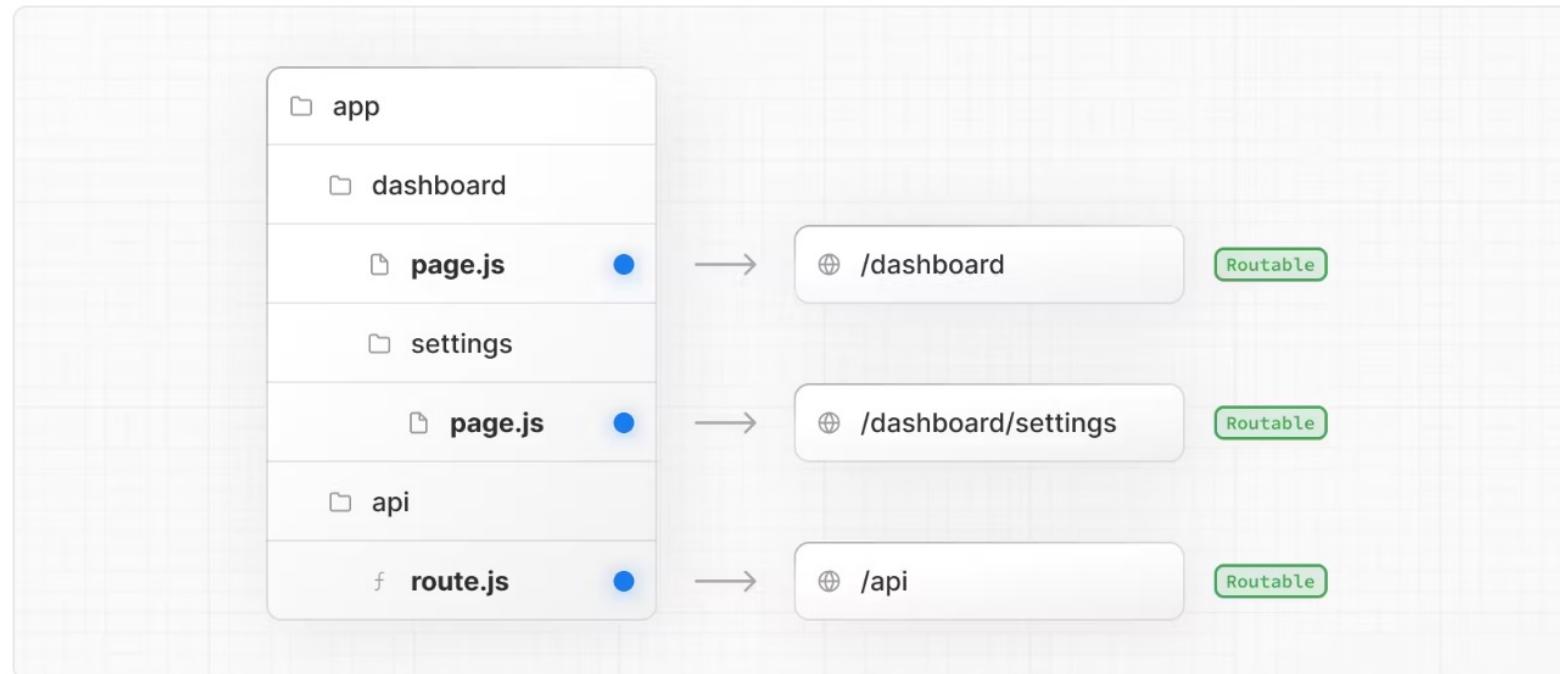


A screenshot of a terminal window showing Prisma migration code. The terminal title is 'prisma > migrations > 20240512000544_init > migration.sql'. The code itself is as follows:

```
1  -- CreateTable
2  CREATE TABLE "Page" (
3      "id" INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
4      "slug" TEXT NOT NULL,
5      "title" TEXT NOT NULL,
6      "content" TEXT NOT NULL
7  );
8
9  -- CreateIndex
10 CREATE UNIQUE INDEX "Page_slug_key" ON "Page"("slug");
11
```

Page based Routing

- **Include a folder within app and it becomes the route**
- **Must have page.js within the folder**



Dynamic Page based Routing

A Dynamic Segment can be created by wrapping a folder's name in square brackets:

`[folderName]`. For example, `[id]` or `[slug]`.

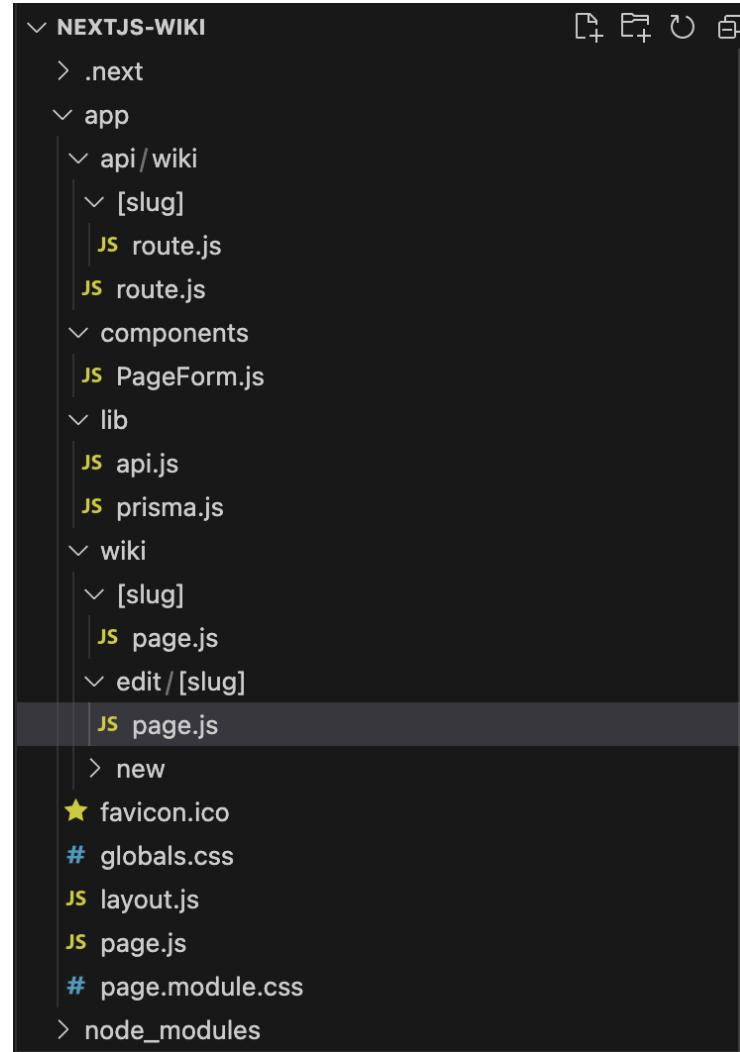
Dynamic Segments are passed as the `params` prop to `layout`, `page`, `route`, and `generateMetadata` functions.

Example

For example, a blog could include the following route `app/blog/[slug]/page.js` where `[slug]` is the Dynamic Segment for blog posts.

```
ts app/blog/[slug]/page.tsx TypeScript ▾  
1 export default function Page({ params }: { params: { slug: string } }) {  
2   return <div>My Post: {params.slug}</div>  
3 }
```

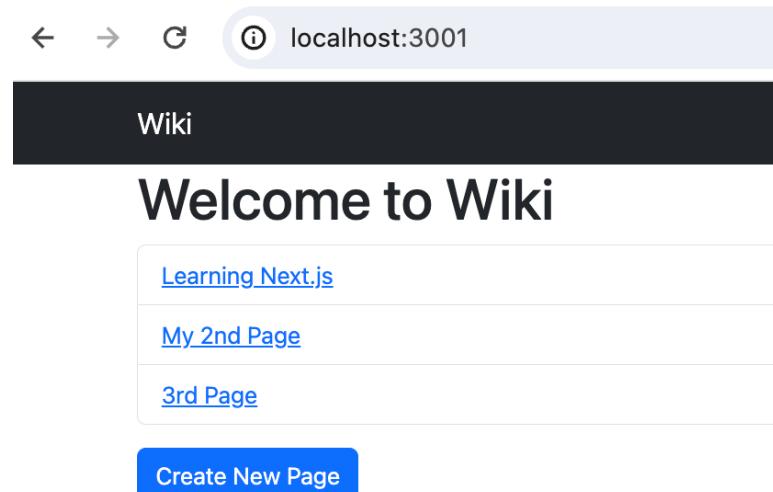
Route	Example URL	params
<code>app/blog/[slug]/page.js</code>	<code>/blog/a</code>	<code>{ slug: 'a' }</code>
<code>app/blog/[slug]/page.js</code>	<code>/blog/b</code>	<code>{ slug: 'b' }</code>
<code>app/blog/[slug]/page.js</code>	<code>/blog/c</code>	<code>{ slug: 'c' }</code>



Run Next.js Dev Server

- Run the dev server
 - npm run dev

Should run on port 3000 unless that is being used. It will then increment the port number eg 3001



Wiki Routes in Action

localhost:3001

Wiki

Welcome to Wiki

- [Learning Next.js](#)
- [My 2nd Page](#)
- [3rd Page](#)

Create New Page

localhost:3001/wiki/new

Wiki

Create New Page

Title:

Content:

Save

localhost:3001/wiki/edit/learning-next.js

Wiki

Edit Page: Learning Next.js

Title:

Content:

```
- This is my first page - I am learning Next.js
```

Save

Dynamic route [slug]

Layout

```
app > JS layout.js > 📦 RootLayout
1   import 'bootstrap/dist/css/bootstrap.css';
2   import Link from 'next/link';
3
4   export default function RootLayout({ children }) {
5       return (
6           <html lang="en">
7               <body>
8                   <nav className="navbar navbar-expand-lg navbar-dark bg-dark">
9                       <div className="container">
10                          <Link href="/" className="navbar-brand">
11                              | Wiki
12                          </Link>
13                          </div>
14                      </nav>
15                      <main>{children}</main>
16                  </body>
17              </html>
18      );
19 }
```

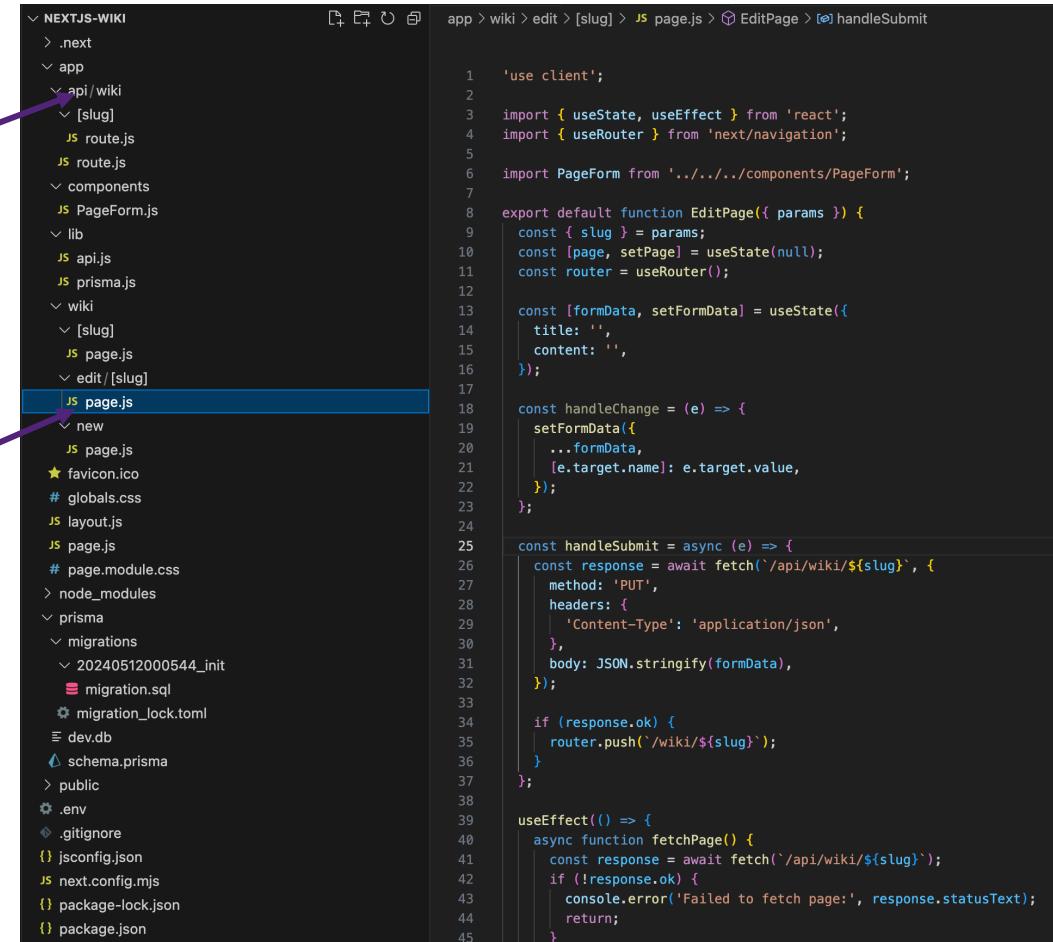
Client-side & Server-side Code

Api runs on server
By default js and React code runs on server

Include 'use client' to run within Web Browser

Example API call:

← → ⌂ ⓘ localhost:3001/api/wiki/learning-next.js



The screenshot shows a code editor with a file tree on the left and a code editor window on the right.

File Tree:

- NEXTJS-WIKI
- .next
- app
- api/wiki
- [slug]
- route.js
- route.js
- components
- PageForm.js
- lib
- api.js
- prisma.js
- wiki
- [slug]
- page.js
- edit/[slug]
- page.js
- new
- page.js
- favicon.ico
- # globals.css
- layout.js
- page.js
- # page.module.css
- node_modules
- prisma
- migrations
- 20240512000544_init
- migration.sql
- migration_lock.toml
- dev.db
- schema.prisma
- public
- .env
- .gitignore
- jsconfig.json
- next.config.mjs
- package-lock.json
- package.json

Code Editor (page.js):

```
1 'use client';
2
3 import { useState, useEffect } from 'react';
4 import { useRouter } from 'next/navigation';
5
6 import PageForm from '../../../../../components/PageForm';
7
8 export default function EditPage({ params }) {
9   const { slug } = params;
10  const [page, setPage] = useState(null);
11  const router = useRouter();
12
13  const [formData, setFormData] = useState({
14    title: '',
15    content: '',
16  });
17
18  const handleChange = (e) => {
19    setFormData({
20      ...formData,
21      [e.target.name]: e.target.value,
22    });
23  };
24
25  const handleSubmit = async (e) => {
26    const response = await fetch(`/api/wiki/${slug}`, {
27      method: 'PUT',
28      headers: {
29        'Content-Type': 'application/json',
30      },
31      body: JSON.stringify(formData),
32    });
33
34    if (response.ok) {
35      router.push(`/wiki/${slug}`);
36    }
37
38  };
39
40  useEffect(() => {
41    async function fetchPage() {
42      const response = await fetch(`/api/wiki/${slug}`);
43      if (!response.ok) {
44        console.error('Failed to fetch page:', response.statusText);
45        return;
46      }
47    }
48  }, []);
49}
```

Other Frameworks

Flask



The screenshot shows the Flask documentation website at flask.palletsprojects.com/en/3.0.x/. The page features a large teal logo icon of a flask on the left, followed by the word "Flask" in a large, bold, black sans-serif font. To the left of the logo is a sidebar with "Project Links" and "Contents" sections, and a "Quick search" bar. Below the sidebar is an advertisement for an on-demand webinar about reducing reliance on bad open source packages.

Project Links

- [Donate](#)
- [PyPI Releases](#)
- [Source Code](#)
- [Issue Tracker](#)
- [Chat](#)

Contents

- [Welcome to Flask](#)
- [User's Guide](#)
- [API Reference](#)
- [Additional Notes](#)

Quick search

 Go

ON-DEMAND WEBINAR
How to reduce your organization's reliance on "bad" open source packages
[WATCH NOW](#) TIDELIFT

On-demand webinar:
Reduce your organization's reliance on "bad" open source packages. **WATCH NOW!**

Ad by EthicalAds

User's Guide

Flask provides configuration and conventions, with sensible defaults, to get started. This section of the documentation explains the different parts of the Flask framework and how they can be used, customized, and extended. Beyond Flask itself, look for community-maintained extensions to add even more functionality.

- [Installation](#)
 - [Python Version](#)
 - [Dependencies](#)
 - [Virtual environments](#)
 - [Install Flask](#)
- [Quickstart](#)
 - [A Minimal Application](#)
 - [Debug Mode](#)
 - [HTML Escaping](#)

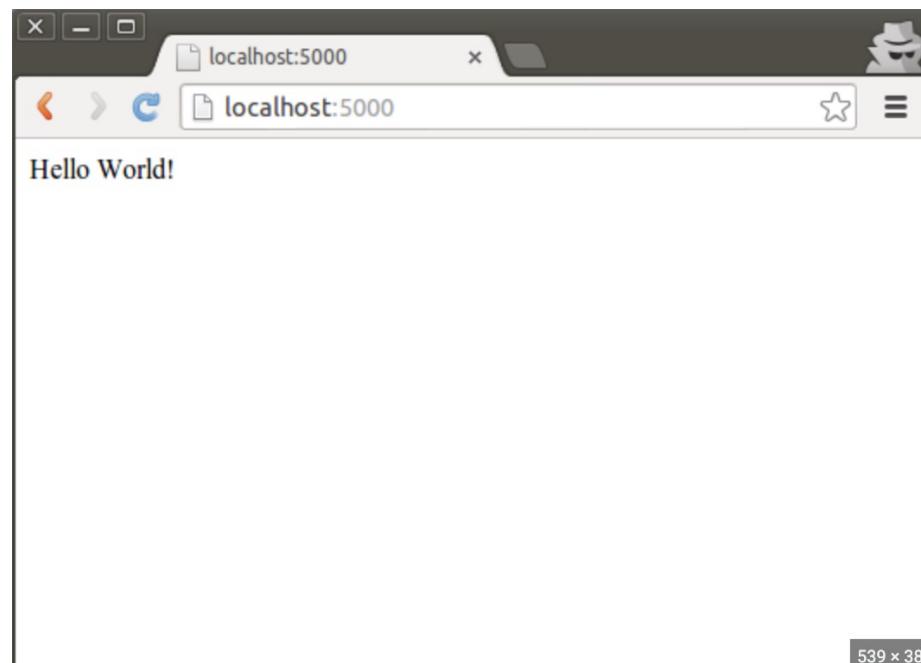
Flask

- Very concise syntax
- Uses decorators in Python to define Routes
- Need to install your own ORM eg SQLAlchemy

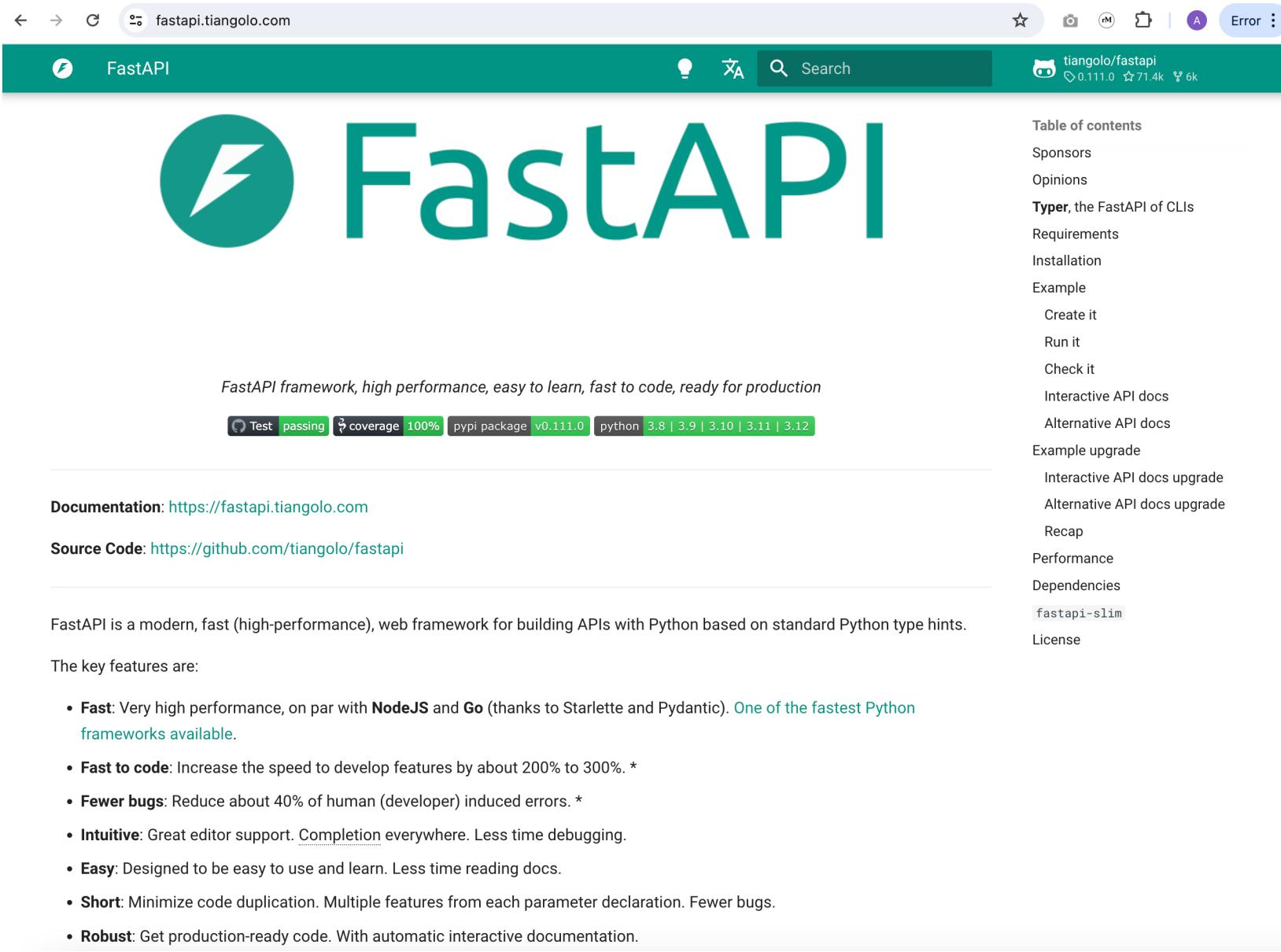
```
from flask import Flask
app = Flask(__name__)

@app.route("/")
def hello() -> str:
    return "Hello World"

if __name__ == "__main__":
    app.run(debug=False)
```



FastAPI



The screenshot shows the official FastAPI documentation website at fastapi.tiangolo.com. The page features a large teal header with the FastAPI logo and navigation links. The main content area includes the FastAPI logo, a large teal title, and a brief introduction. A sidebar on the right lists various documentation sections.

Header: fastapi.tiangolo.com

Page Title: FastAPI

Page Description: FastAPI framework, high performance, easy to learn, fast to code, ready for production

Test Status: Test passing

Code Coverage: coverage 100%

Pypi Package: pypi package v0.111.0

Python Versions: python 3.8 | 3.9 | 3.10 | 3.11 | 3.12

Documentation: <https://fastapi.tiangolo.com>

Source Code: <https://github.com/tiangolo/fastapi>

FastAPI is a modern, fast (high-performance), web framework for building APIs with Python based on standard Python type hints.

The key features are:

- **Fast:** Very high performance, on par with **NodeJS** and **Go** (thanks to Starlette and Pydantic). [One of the fastest Python frameworks available.](#)
- **Fast to code:** Increase the speed to develop features by about 200% to 300%. *
- **Fewer bugs:** Reduce about 40% of human (developer) induced errors. *
- **Intuitive:** Great editor support. [Completion everywhere.](#) Less time debugging.
- **Easy:** Designed to be easy to use and learn. Less time reading docs.
- **Short:** Minimize code duplication. Multiple features from each parameter declaration. Fewer bugs.
- **Robust:** Get production-ready code. With automatic interactive documentation.

Table of contents:

- Sponsors
- Opinions
- Typer**, the FastAPI of CLIs
- Requirements
- Installation
- Example
 - Create it
 - Run it
 - Check it
 - Interactive API docs
 - Alternative API docs
- Example upgrade
 - Interactive API docs upgrade
 - Alternative API docs upgrade
- Recap
- Performance
- Dependencies
- fastapi-slim**
- License

SECaTs for Semester 1, 2024

13 May – 31 May (11:59 pm AEST)

What are the benefits of completing the survey?

The SECaT is your opportunity to provide feedback on your learning experience!

Your responses will be used to improve courses and teaching for future students. Please be assured that all responses are strictly confidential.

How do I complete the survey?

Visit <https://eval.uq.edu.au/eus.onlinesurveyportal/> or scan this QR code:



Please provide specific examples where possible. Respectful constructive feedback is always helpful to help improve your learning experience.



Week 12: Todo

- Continue with your Project Assessment Item
- Project Assessment Item is due Friday 17 May at 3pm
- In the Lab you can work on your project
- If you don't have a Student Card, please get one!
- You will need a Student Card for the Exam



CREATE CHANGE

Thank you