Express.js

- What is Express.js?
- How to use Express.js?
  - SSR
  - API

What is Express.js?

- Express.js is a fast, un-opinionated, minimalist web application framework for Node.js
- Minimalist: Express.js is a lightweight framework. It doesn't have a lot of built-in features. Instead, it provides a lot of flexibility to developers to add features as per their requirements.
- Un-opinionated: Express.js doesn't force you to use any specific ORM or template engine. It gives you the flexibility to use any database and template engine that you want.
- https://expressjs.com/

Why Express.js?

- It's un-opinionated
- It's very flexible and plugable
- HTML templates - Pug, EJS, Handlebars, etc.
- Database - MongoDB, MySQL, PostgreSQL, etc.

First App with Express.js

```
const express = require('express');
const app = express();

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

app.listen(3000, () => {
  console.log('Example app listening on port 3000!');
});
```

Static Site

static-site.js

```
app.use(express.static('public'));
```

Dynamic Site - Server Side Rendering

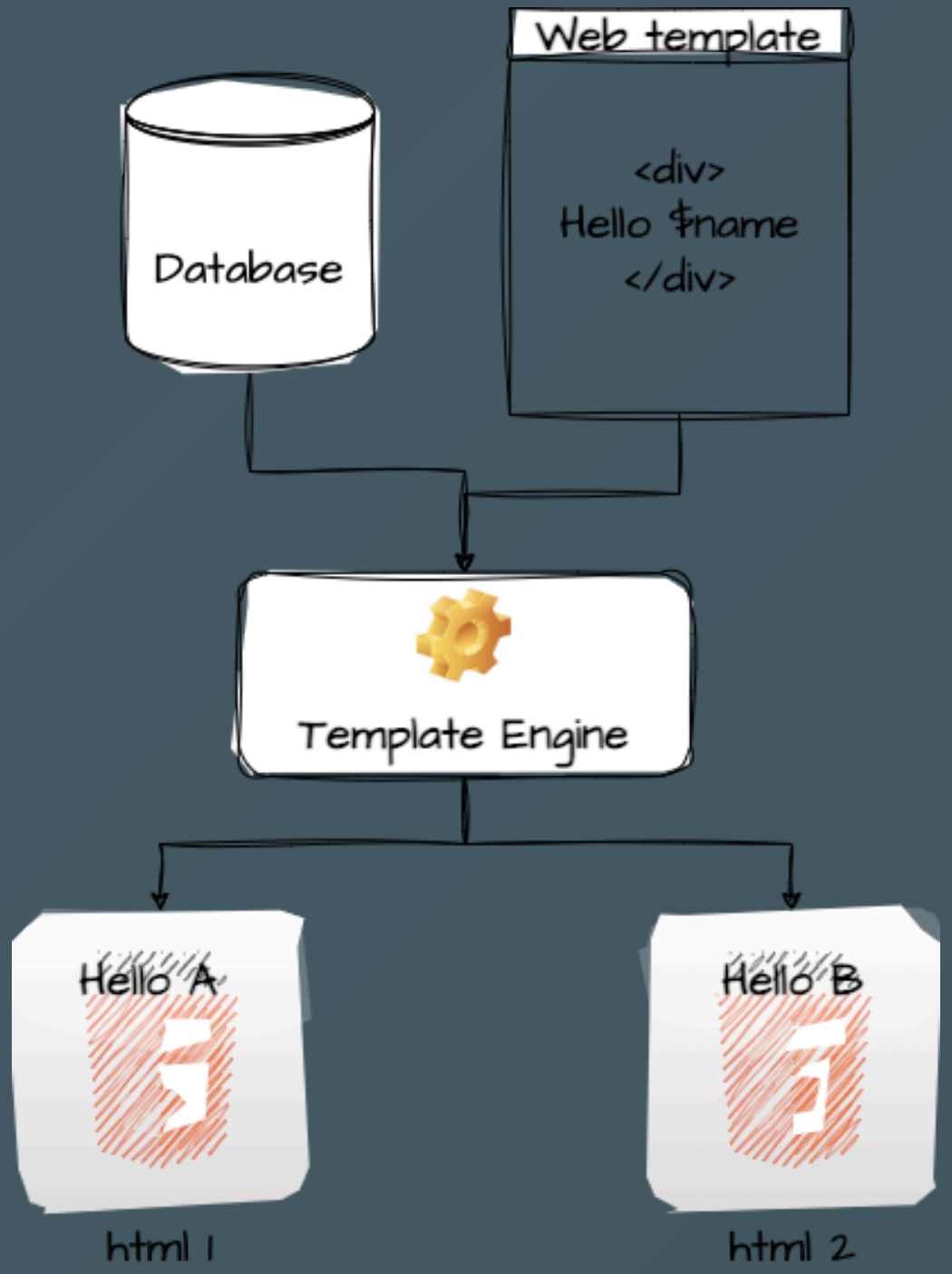
dynamic-site.js

```
app.set('view engine', 'pug');
app.set('views', './views');

app.get('/', (req, res) => {
  res.render('index', { title: 'Hey', message: 'Hello there!' });
});
```

Template Engine

- Generate dynamic HTML pages
- Separating business logic from presentation logic
- Reusable components
- Pug, EJS, Handlebars, etc.



Template Engine - Pug

```
html
  head
    title= title
  body
    h1 #{message}
```

API

- Application Programming Interface
- A set of clearly defined methods of communication between various software components
- REST API - Representational State Transfer (will be covered in the coming lecture)
- JSON

JSON Basics

- JavaScript Object Notation
- A lightweight data-interchange format
- Easy for humans to read and write
- Easy for machines to parse and generate
- JSON is built on two structures:
  - A collection of name/value pairs
  - An ordered list of values
- https://api.github.com/users/chuwa-fullstack-training

Params, Query String

```
app.get('/users/:userId', (req, res) => {
  res.send(req.params.userId);
});

app.get('/users', (req, res) => {
  res.send(req.query);
});
```

Route

route.js

- A route is a section of Express code that associates an HTTP verb (GET, POST, PUT, DELETE, etc.), a URL path/pattern, and a function that is called to handle that pattern.

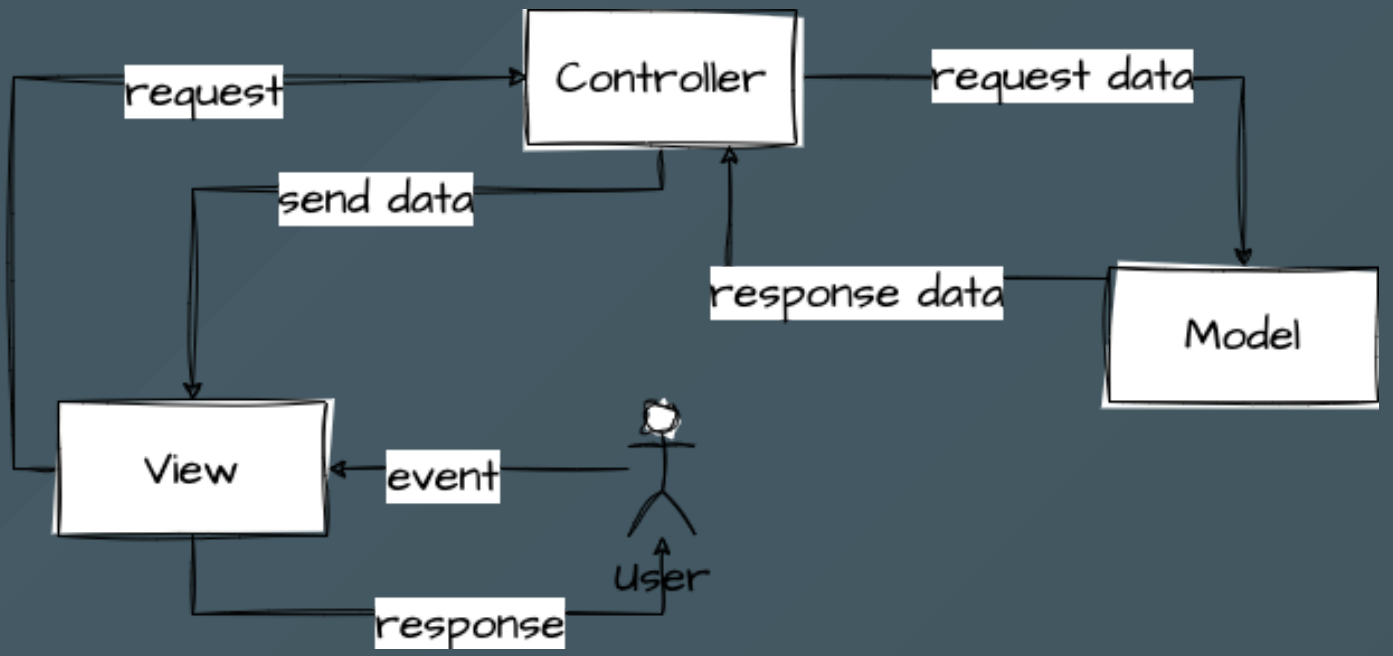
URL Encoding and Decoding

- URL encoding is the practice of translating unprintable characters or characters with special meaning within URLs to a representation that is unambiguous and universally accepted by web browsers and servers.
- Example:
  - `https://www.google.com/search?q=url&encoding`
  - `https://www.google.com/search?q=url%26encoding`
- https://www.albionresearch.com/tools/urlencode#:~:text=URL Encoding is used when,data to a web server

MVC Pattern

mvc-index.js

- Model-View-Controller
- Separating business logic from presentation logic



Middleware

middleware.js

- Middleware functions are functions that have access to the request object (req), the response object (res), and the next middleware function in the application’s request-response cycle.
- Middleware functions can perform the following tasks:
  - Execute any code.
  - Make changes to the request and the response objects.
  - End the request-response cycle.
  - Call the next middleware function in the stack.

Router

router.js

- Router-level middleware works in the same way as application-level middleware, except it is bound to an instance of express.Router().
- A router object is an isolated instance of middleware and routes. You can think of it as a “mini-application,” capable only of performing middleware and routing functions. Every Express application has a built-in app router.

`app` object

- `app.get()`
- `app.post()`
- `app.put()`
- `app.delete()`
- `app.all()`
- `app.use()`
- `app.listen()`

`req` & `res` objects

- `req.params`
- `req.query`
- `req.body`
- `req.headers`
- `req.cookies`

- `res.send()`
- `res.json()`