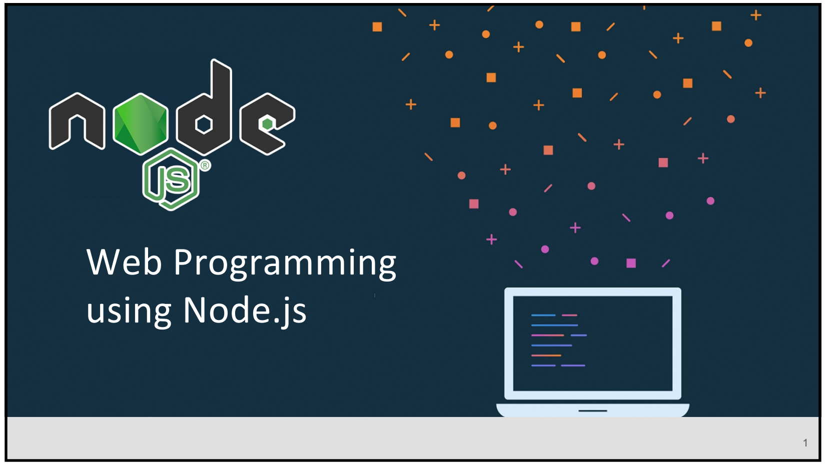
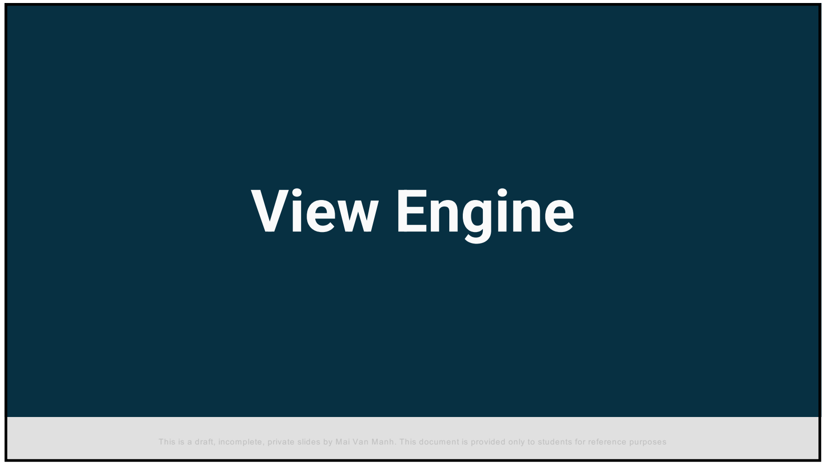


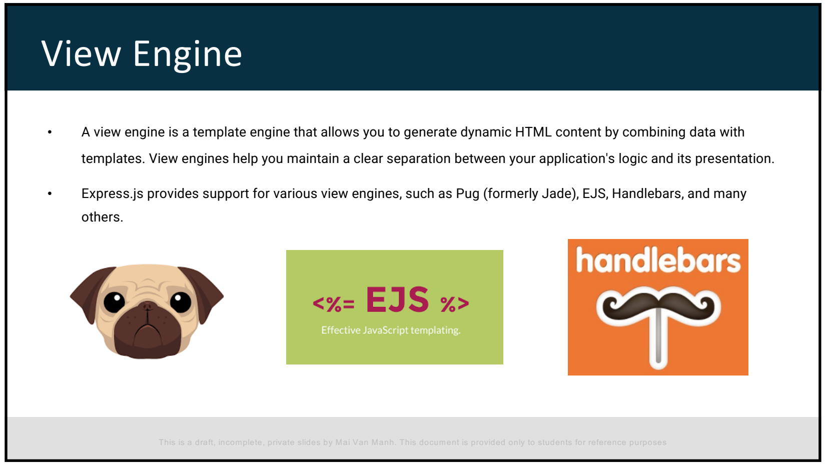
Web Programming using Node.js

1



View Engine

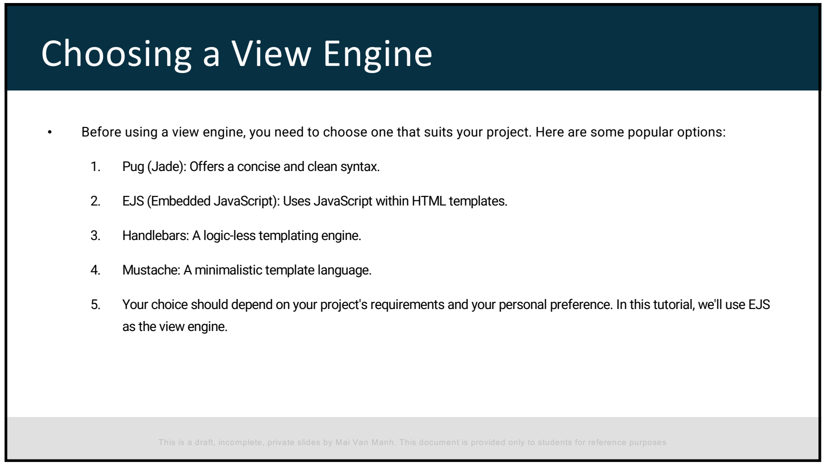
2



View Engine

- A view engine is a template engine that allows you to generate dynamic HTML content by combining data with templates. View engines help you maintain a clear separation between your application's logic and its presentation.
- Express.js provides support for various view engines, such as Pug (formerly Jade), EJS, Handlebars, and many others.

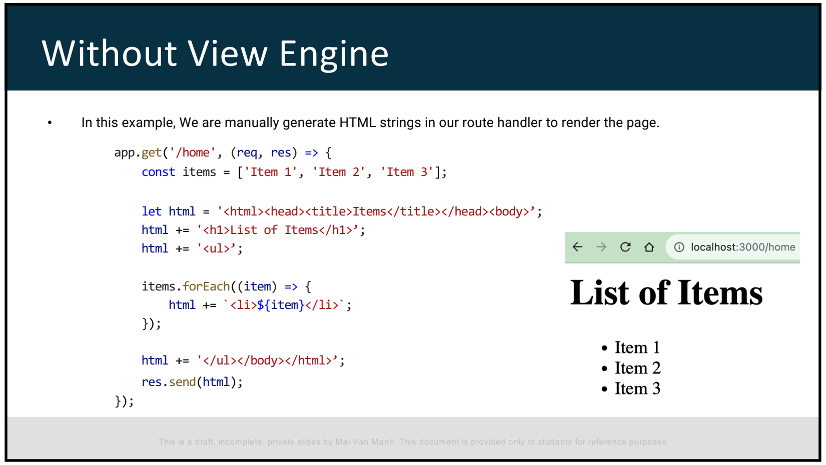
3



Choosing a View Engine

- Before using a view engine, you need to choose one that suits your project. Here are some popular options:
  1. Pug (Jade): Offers a concise and clean syntax.
  2. EJS (Embedded JavaScript): Uses JavaScript within HTML templates.
  3. Handlebars: A logic-less templating engine.
  4. Mustache: A minimalistic template language.
  5. Your choice should depend on your project's requirements and your personal preference. In this tutorial, we'll use EJS as the view engine.

4



Without View Engine

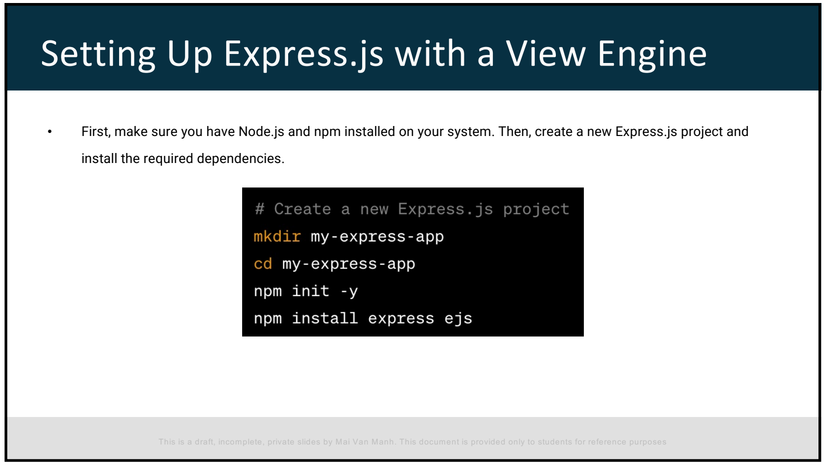
- In this example, We are manually generate HTML strings in our route handler to render the page.

```
app.get('/home', (req, res) => {
    const items = ['Item 1', 'Item 2', 'Item 3'];

    let html = '<html><head><title>Items</title></head><body>’;
    html += '<h1>List of Items</h1>’;
    html += '<ul>’;

    items.forEach((item) => {
        html += `<li>${item}</li>`;
    });

    html += '</ul></body></html>’;
    res.send(html);
});
```

List of Items

- Item 1
- Item 2
- Item 3

5

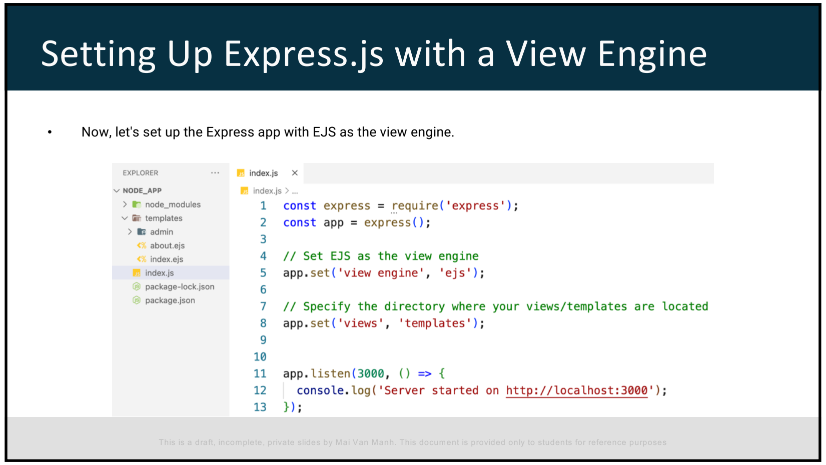


Setting Up Express.js with a View Engine

- First, make sure you have Node.js and npm installed on your system. Then, create a new Express.js project and install the required dependencies.

```
# Create a new Express.js project
mkdir my-express-app
cd my-express-app
npm init -y
npm install express ejs
```

6

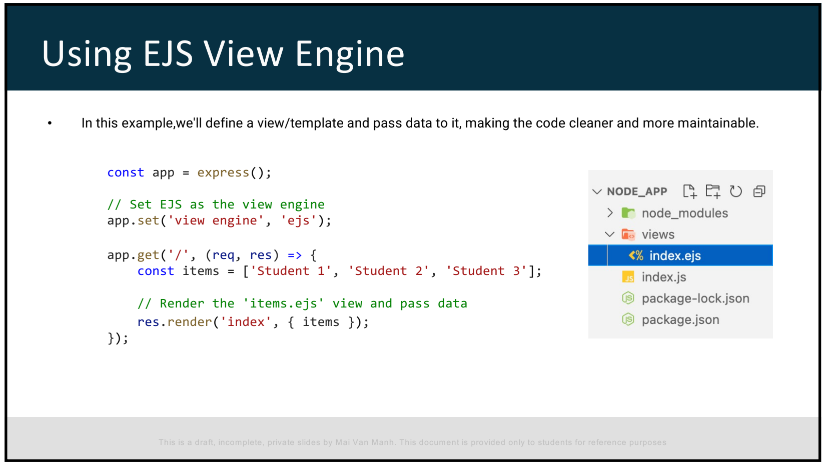


Setting Up Express.js with a View Engine

- Now, let's set up the Express app with EJS as the view engine.

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

// Set EJS as the view engine
app.set('view engine', 'ejs');

// Specify the directory where your views/templates are located
app.set('views', 'templates');


app.listen(3000, () => {
  console.log('Server started on http://localhost:3000');
});
```

7



Using EJS View Engine

- In this example,we'll define a view/template and pass data to it, making the code cleaner and more maintainable.

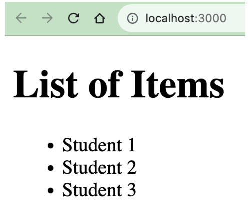
```
const app = express();

// Set EJS as the view engine
app.set('view engine', 'ejs');

app.get('/', (req, res) => {
    const items = ['Student 1', 'Student 2', 'Student 3'];

    // Render the 'items.ejs' view and pass data
    res.render('index', { items });
});
```

8

Using EJS View Engine

- In this example,we'll define a view/template and pass data to it, making the code cleaner and more maintainable.

```
<!DOCTYPE html>
    <html>
        <head>
            <title>Items</title>
        </head>
        <body>
            <h1>List of Items</h1>
            <ul>
                <% items.forEach((item) => { %>
                    <li><%= item %></li>
                <% }); %>
            </ul>
    </body>
</html>
```



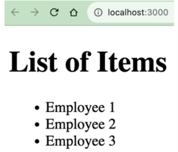
List of Items

- Student 1
- Student 2
- Student 3

Using EJS View Engine

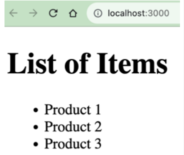
- Using a view engine increases flexibility, we can easily dump new data into the interface.

```
app.get('/', (req, res) => {
    const items = ['Employee 1', 'Employee 2', 'Employee 3’];
    res.render('index', { items });
});


app.get('/', (req, res) => {
    const items = [Product 1', 'Product 2', ‘Product 3’];
    res.render('index', { items });
});
```



List of Items

- Employee 1
- Employee 2
- Employee 3



List of Items

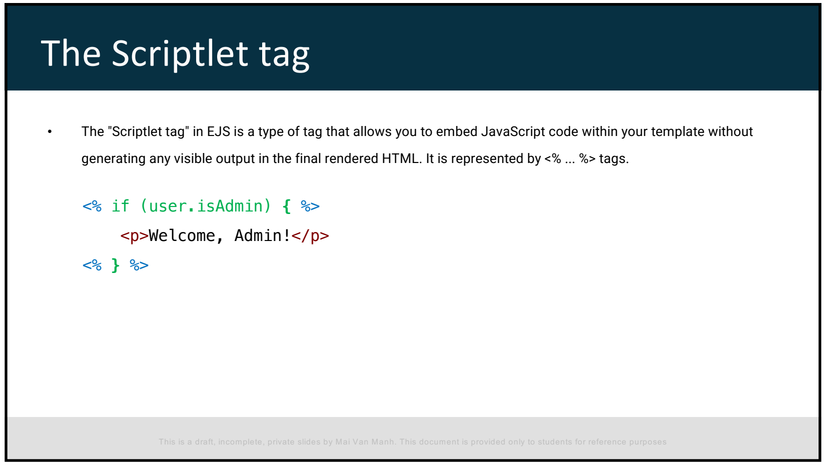
- Product 1
- Product 2
- Product 3

View Engine Benefits

- View engines offer several benefits for Express.js developers:
    - Separation of concerns: View engines allow developers to separate the presentation layer of their application from the logic layer. This makes it easier to maintain and update the application.
    - Reusability: View engines allow developers to create reusable templates that can be used across multiple pages in an application. This can save a lot of time and effort.
    - Dynamic content: View engines allow developers to generate dynamic content for their web pages. This means that the content of a page can change depending on the user, the data in the database, or other factors.

EJS View Tags

- <% 'Scriptlet' tag, for control-flow, no output
- <%= Outputs the value into the template (HTML escaped)
- <%- Outputs the unescaped value into the template
- <%# Comment tag, no execution, no output
- <%% Outputs a literal '<%'
- %> Plain ending tag

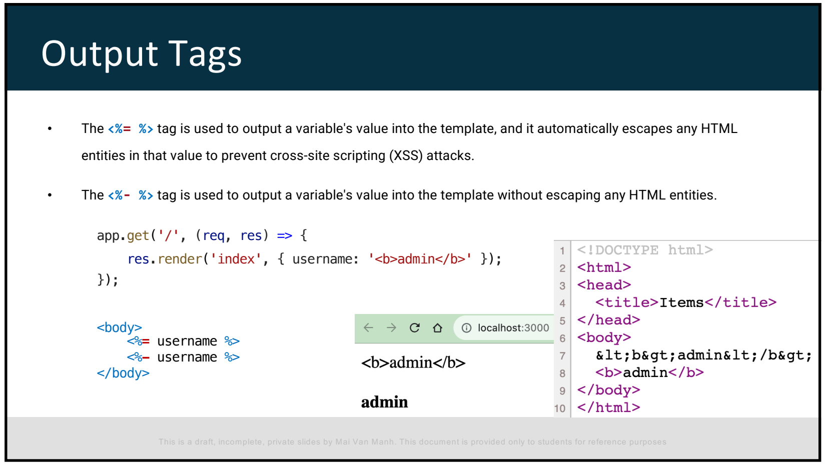


The Scriptlet tag

- The "Scriptlet tag" in EJS is a type of tag that allows you to embed JavaScript code within your template without generating any visible output in the final rendered HTML. It is represented by <% ... %> tags.

```
<% if (user.isAdmin) { %>
    <p>Welcome, Admin!</p>
<% } %>
```

13



Output Tags

- The <%= %> tag is used to output a variable's value into the template, and it automatically escapes any HTML entities in that value to prevent cross-site scripting (XSS) attacks.
- The <%- %> tag is used to output a variable's value into the template without escaping any HTML entities.

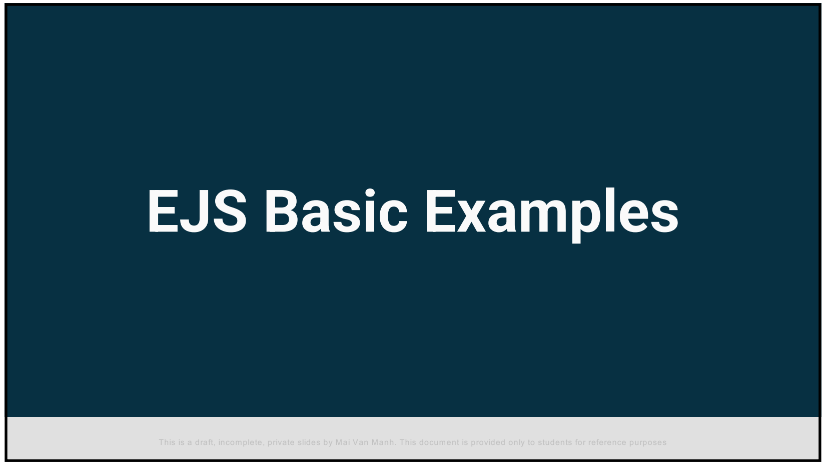
```
app.get('/', (req, res) => {
    res.render('index', { username: '<b>admin</b>' });
});
```

```
<body>
    <%= username %>
    <%- username %>
</body>
```

```
<b>admin</b>

admin
```

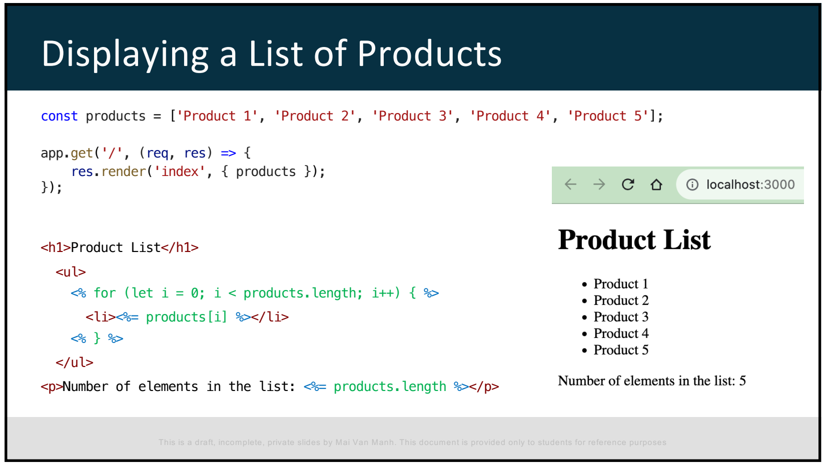
```
<!DOCTYPE html>
<html>
<head>
  <title>Items</title>
</head>
<body>
  &lt;b&gt;admin&lt;/b&gt;
  <b>admin</b>
</body>
</html>
```

14



EJS Basic Examples

15



Displaying a List of Products

```
const products = ['Product 1', 'Product 2', 'Product 3', 'Product 4', 'Product 5'];

app.get('/', (req, res) => {
    res.render('index', { products });
});
```

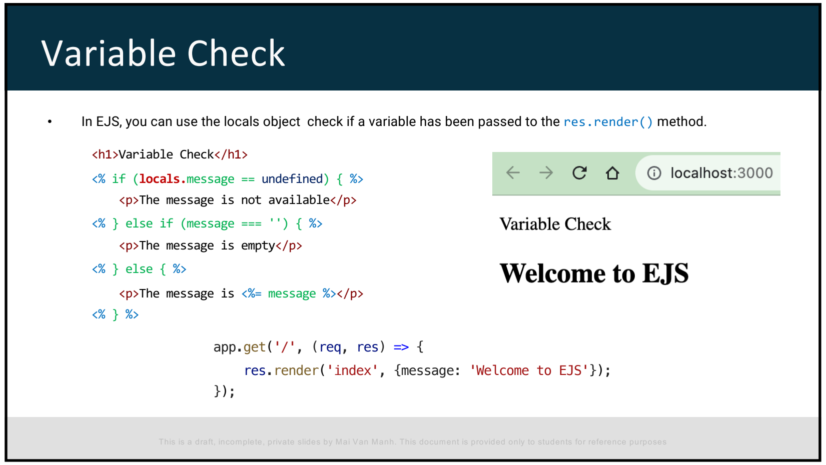
```
<h1>Product List</h1>
  <ul>
    <% for (let i = 0; i < products.length; i++) { %>
      <li><%= products[i] %></li>
    <% } %>
  </ul>
<p>Number of elements in the list: <%= products.length %></p>
```

Product List

- Product 1
- Product 2
- Product 3
- Product 4
- Product 5

Number of elements in the list: 5

16



Variable Check

- In EJS, you can use the locals object check if a variable has been passed to the `res.render()` method.

```
<h1>Variable Check</h1>
<% if (locals.message == undefined) { %>
    <p>The message is not available</p>
<% } else if (message === '') { %>
    <p>The message is empty</p>
<% } else { %>
    <p>The message is <%= message %></p>
<% } %>
```

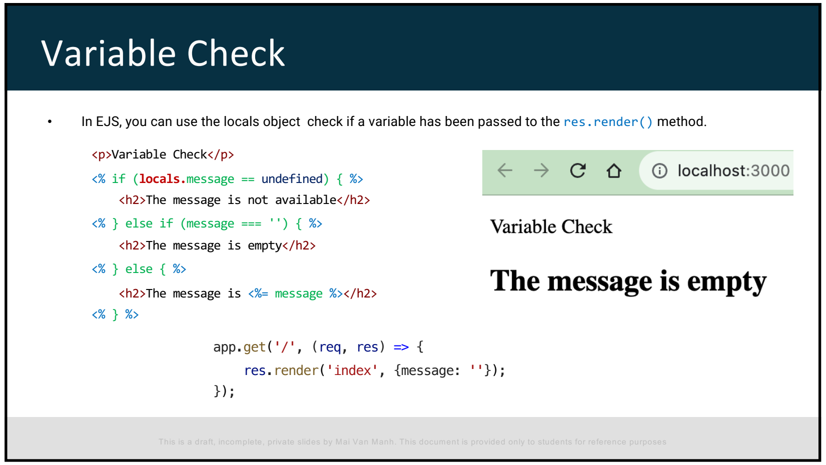
localhost:3000

Variable Check

Welcome to EJS

```
app.get('/', (req, res) => {
    res.render('index', {message: 'Welcome to EJS'});
});
```

17



Variable Check

- In EJS, you can use the locals object check if a variable has been passed to the `res.render()` method.

```
<p>Variable Check</p>
<% if (locals.message == undefined) { %>
    <h2>The message is not available</h2>
<% } else if (message === '') { %>
    <h2>The message is empty</h2>
<% } else { %>
    <h2>The message is <%= message %></h2>
<% } %>
```

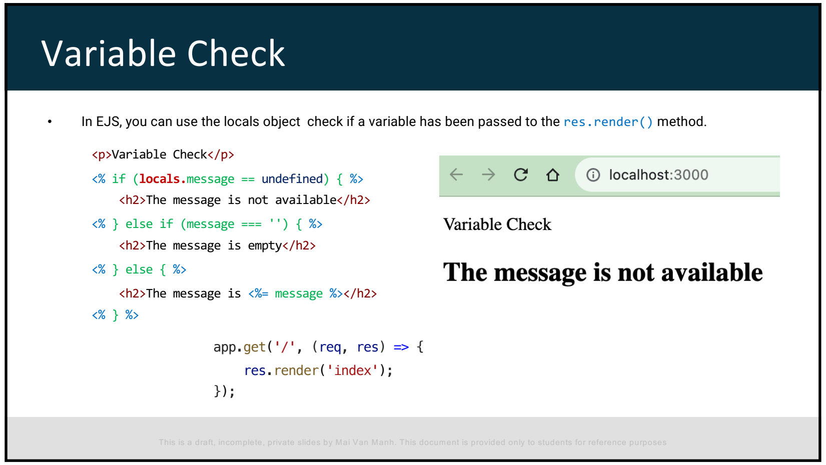
localhost:3000

Variable Check

The message is empty

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

18



Variable Check

- In EJS, you can use the locals object check if a variable has been passed to the `res.render()` method.

```
<p>Variable Check</p>
<% if (locals.message == undefined) { %>
    <h2>The message is not available</h2>
<% } else if (message === '') { %>
    <h2>The message is empty</h2>
<% } else { %>
    <h2>The message is <%= message %></h2>
<% } %>
```

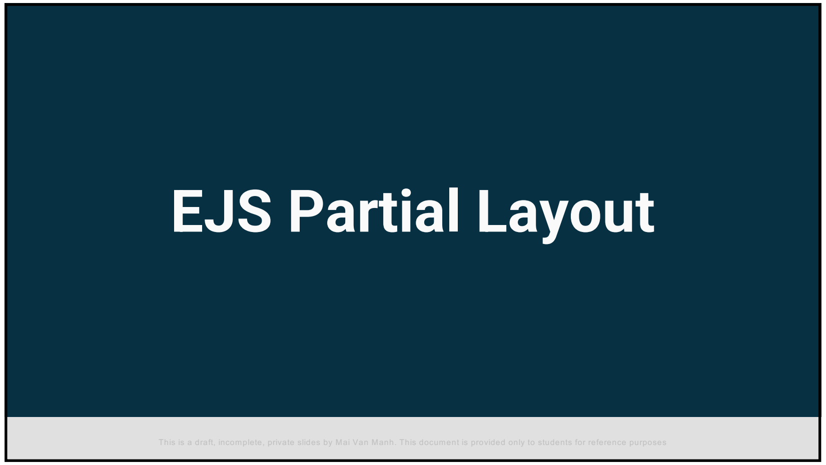
localhost:3000

Variable Check

The message is not available

```
app.get('/', (req, res) => {
    res.render('index');
});
```

19



EJS Partial Layout

20

Partial Layout

- In EJS (Embedded JavaScript) templates, you can create a master layout by defining a template that contains the common structure and elements you want to reuse across multiple pages.
- Organize your project with a directory structure that separates your views, layouts, and partials. Here's a common structure:

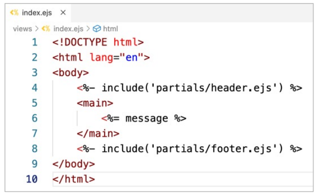
```
my-ejs-project/
├── views/
│ ├── partials/
│ │ ├── header.ejs          <-- Header Partial
│ │ └── footer.ejs          <-- Footer Partial
│ └── main.ejs              <-- Main layout
├── app.js
└── package.json
```

This is a draft, incomplete, private slides by Mai Van Manh. This document is provided only to students for reference purposes

21

Partial Layout

```
my-ejs-project/
├── views/
│ ├── partials/
│ │ ├── header.ejs
│ │ └── footer.ejs
│ └── main.ejs
├── app.js
└── package.json
```

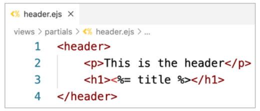


index.ejs ×

views > index.ejs > html

```
<!DOCTYPE html>
<html lang="en">
<body>
    <%- include('partials/header.ejs') %>
    <main>
        <%= message %>
    </main>
    <%- include('partials/footer.ejs') %>
</body>
</html>
```

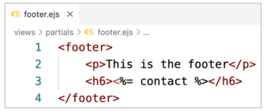
```
app.get('/', (req, res) => {
  res.render('index', {
    message: 'Main Content',
    title: 'EJS Include Example',
    contact: 'abc@gmail.com'});
});
```



header.ejs ×

views > partials > header.ejs > ...

```
<header>
    <p>This is the header</p>
    <h1><%= title %></h1>
</header>
```



footer.ejs ×

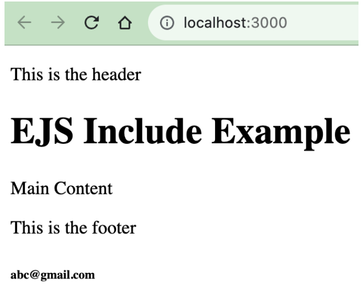
views > partials > footer.ejs > ...

```
<footer>
    <p>This is the footer</p>
    <h6><%= contact %></h6>
</footer>
```

This is a draft, incomplete, private slides by Mai Van Manh. This document is provided only to students for reference purposes

22

Partial Layout



localhost:3000

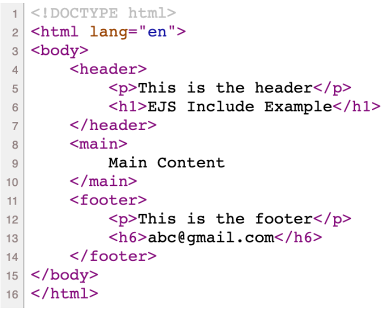
This is the header

EJS Include Example

Main Content

This is the footer

abc@gmail.com



```
<!DOCTYPE html>
<html lang="en">
<body>
    <header>
        <p>This is the header</p>
        <h1>EJS Include Example</h1>
    </header>
    <main>
        Main Content
    </main>
    <footer>
        <p>This is the footer</p>
        <h6>abc@gmail.com</h6>
    </footer>
</body>
</html>
```

This is a draft, incomplete, private slides by Mai Van Manh. This document is provided only to students for reference purposes

23

express-ejs-layouts

This is a draft, incomplete, private slides by Mai Van Manh. This document is provided only to students for reference purposes

24

Master Layout

- This library allows you to define a common layout structure for your web pages and insert content dynamically.

```
npm install express-ejs-layouts
```

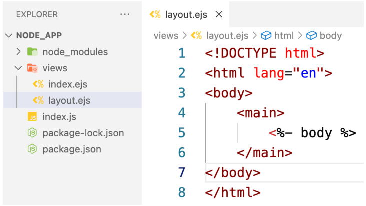
- After installation, you should require and configure the express-ejs-layouts middleware in your app.js file, just before you define your routes.

```
const app = express();
app.set('view engine', 'ejs’);

const expressLayouts = require('express-ejs-layouts');
app.use(expressLayouts);
```

25

Master Layout

- Create a folder named views in your project root directory (if it doesn't exist already). Inside this folder, create an EJS layout file, for example, layout.ejs. This file will serve as the common layout structure for your web pages.

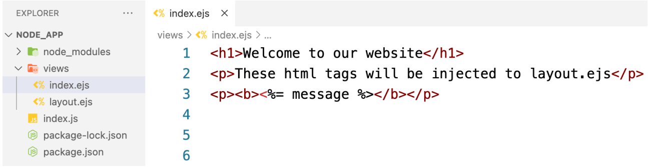


```
<!DOCTYPE html>
<html lang="en">
<body>
    <main>
        <%- body %>
    </main>
</body>
</html>
```

26

Master Layout

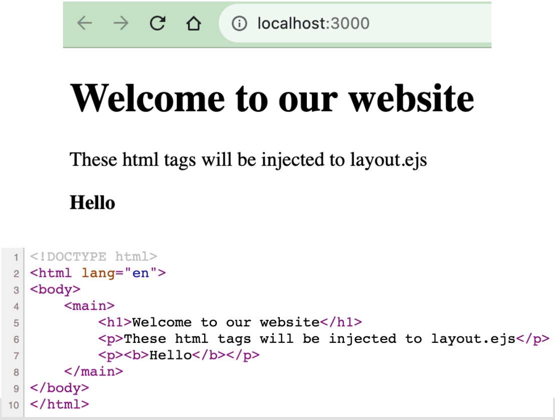
- Now, you can create individual EJS views that will be injected into the layout. For example, create a views/index.ejs



```
<h1>Welcome to our website</h1>
<p>These html tags will be injected to layout.ejs</p>
<p><b><%= message %></b></p>
```

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

27



Welcome to our website

These html tags will be injected to layout.ejs

Hello

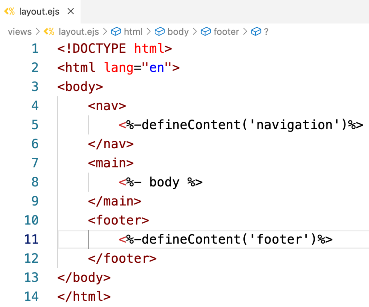
```
<!DOCTYPE html>
<html lang="en">
<body>
    <main>
        <h1>Welcome to our website</h1>
        <p>These html tags will be injected to layout.ejs</p>
        <p><b>Hello</b></p>
    </main>
</body>
</html>
```

28

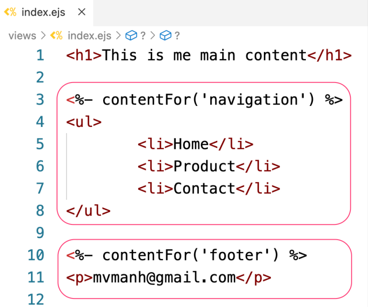
29

Master Layout

- In a layout, you can have optional sections using defineContent. In child views, we use contentFor to define sections.



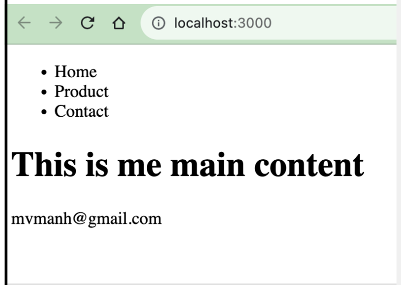
```
<!DOCTYPE html>
<html lang="en">
<body>
    <nav>
        <%-defineContent('navigation')%>
    </nav>
    <main>
        <%- body %>
    </main>
    <footer>
        <%-defineContent('footer')%>
    </footer>
</body>
</html>
```



```
<h1>This is me main content</h1>

<%- contentFor('navigation') %>
<ul>
        <li>Home</li>
        <li>Product</li>
        <li>Contact</li>
</ul>

<%- contentFor('footer') %>
<p>mvmanh@gmail.com</p>

```

30

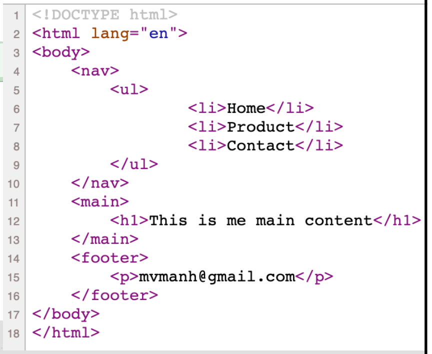
Master Layout



- Home
- Product
- Contact

This is me main content

mvmanh@gmail.com



```
<!DOCTYPE html>
<html lang="en">
<body>
    <nav>
        <ul>
                <li>Home</li>
                <li>Product</li>
                <li>Contact</li>
        </ul>
    </nav>
    <main>
        <h1>This is me main content</h1>
    </main>
    <footer>
        <p>mvmanh@gmail.com</p>
    </footer>
</body>
</html>
```

31

Form Handling Example With Express

32