

Feb 23 / 2023
wN6

NodeJS Mustache

Eko Kurniawan Khannedy

Eko Kurniawan Khannedy

- Technical architect at one of the biggest ecommerce company in Indonesia
- 12+ years experiences
- www.programmerzamannow.com
- youtube.com/c/ProgrammerZamanNow



Eko Kurniawan Khannedy

- Telegram : [@khannedy](https://t.me/khannedy)
- Facebook : [fb.com/ProgrammerZamanNow](https://www.facebook.com/ProgrammerZamanNow)
- Instagram : [instagram.com/programmerzamannow](https://www.instagram.com/programmerzamannow)
- Youtube : [youtube.com/c/ProgrammerZamanNow](https://www.youtube.com/c/ProgrammerZamanNow)
- Telegram Channel : t.me/ProgrammerZamanNow
- Tiktok : [https://tiktok.com/@programmerzamannow](https://www.tiktok.com/@programmerzamannow)
- Email : echo.khannedy@gmail.com

Sebelum Belajar

- Sudah Mengikuti Kelas JavaScript Programmer Zaman Now
- NodeJS Dasar, Unit Test, Logging, dan ExpressJS

Agenda

- Pengenalan Mustache
- Tags
- Sections
- Inverted Sections
- Functions
- Partials
- Dan lain-lain

Sekar. 13/09/2023

Pengenalan Mustache

Pengenalan Mustache

- Mustache adalah library untuk Template Engine
- Dengan menggunakan Mustache, kita bisa membuat kode template, yang nanti isi data bisa kita masukkan ke dalam template sehingga menghasilkan konten yang kita mau
- Mustache sangat sederhana, oleh karena itu sangat mudah dipelajari
- Mustache sendiri sudah menjadi standar, sehingga bisa digunakan di banyak teknologi dan bahasa pemrograman
- <https://mustache.github.io/>
- Semua standarisasi nya bisa dilihat di <https://mustache.github.io/mustache.5.html>

NodeJS Mustache

- Salah satu implementasi dari Mustache di NodeJS adalah library **MustacheJS**
- <https://github.com/janl/mustache.js/>
- <https://www.npmjs.com/package/mustache>

Membuat Project

Membuat Project

- Clone project belajar-nodejs-unit-test
- <https://github.com/ProgrammerZamanNow/belajar-nodejs-unit-test>



Menambah Dependency

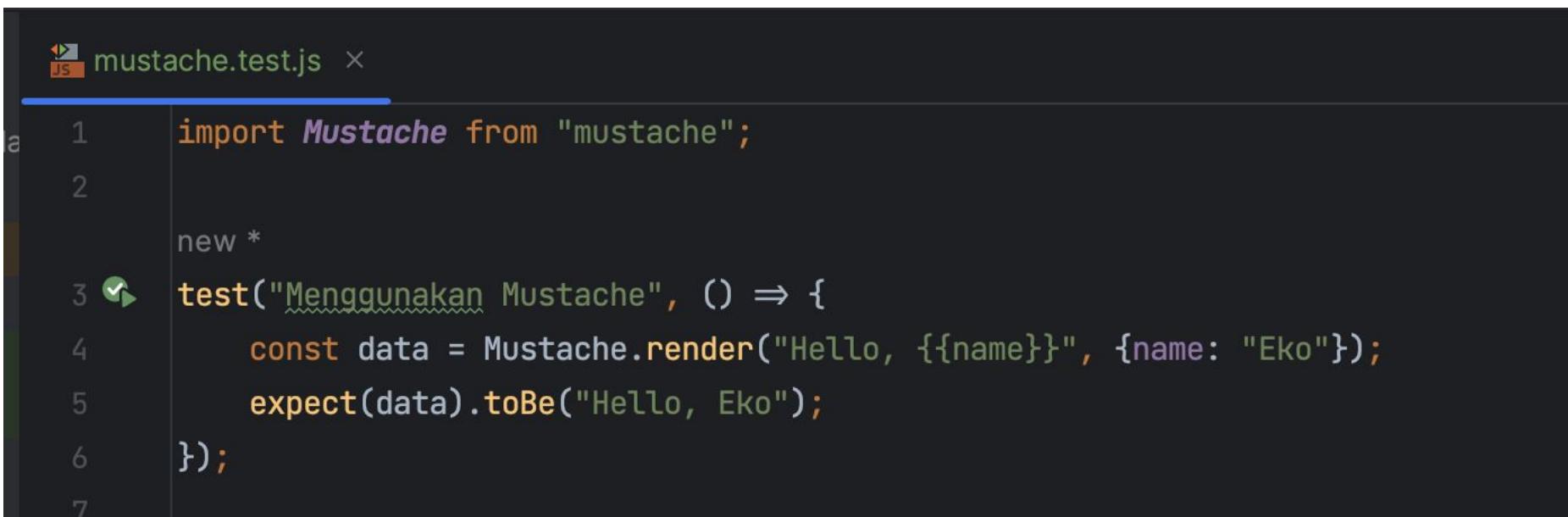
```
30     "author": "Eko Kurniawan Khannedy",
31     "license": "ISC",
32     "dependencies": {
33       "mustache": "^4.2.0"
34     },
35     "devDependencies": {
36       "@babel/plugin-transform-runtime": "^7.17.12",
37       "@babel/preset-env": "^7.17.12",
38       "babel-jest": "^28.1.0",
39       "jest": "^28.1.0",
40       "@types/jest": "^28.1.0",
41       "@types/mustache": "^4.2.0"
42     }
43   }
```

Menggunakan Mustache JS

Menggunakan Mustache JS

- Sebelum kita belajar tentang Mustache Template Engine, kita akan belajar bagaimana menggunakan Mustache JS
- Untuk menggunakan Mustache JS, terdapat function bernama render(template, data) di package mustache, kita bisa gunakan untuk merender template dan data

Kode : Menggunakan Mustache JS



The screenshot shows a code editor window with a dark theme. At the top, there is a small horizontal bar consisting of two colored segments: a teal segment on the left and an orange segment on the right.

The code editor displays a file named `mustache.test.js`. The file content is as follows:

```
1 import Mustache from "mustache";
2
3 new *
4
5 test("Menggunakan Mustache", () => {
6     const data = Mustache.render("Hello, {{name}}", {name: "Eko"});
7     expect(data).toBe("Hello, Eko");
8 });
9
```

The code uses the `Mustache` library to render a template string with a placeholder `name` and assert its value using `expect`.

Mustache Cache

Mustache Cache

- Saat kita menggunakan function render(), data template akan di cache oleh Mustache JS, hal ini agar ketika kita melakukan render dengan template yang sama, proses akan menjadi lebih cepat
- Biasanya, proses render di awal akan sedikit lebih lambat, dikarenakan butuh melakukan proses kompilasi template terlebih dahulu
- Jika kita ingin mempercepat, di awal aplikasi berjalan, kita bisa lakukan kompilasi semua template terlebih dahulu, dengan cara menggunakan function parse(template)

Kode : Mustache Cache

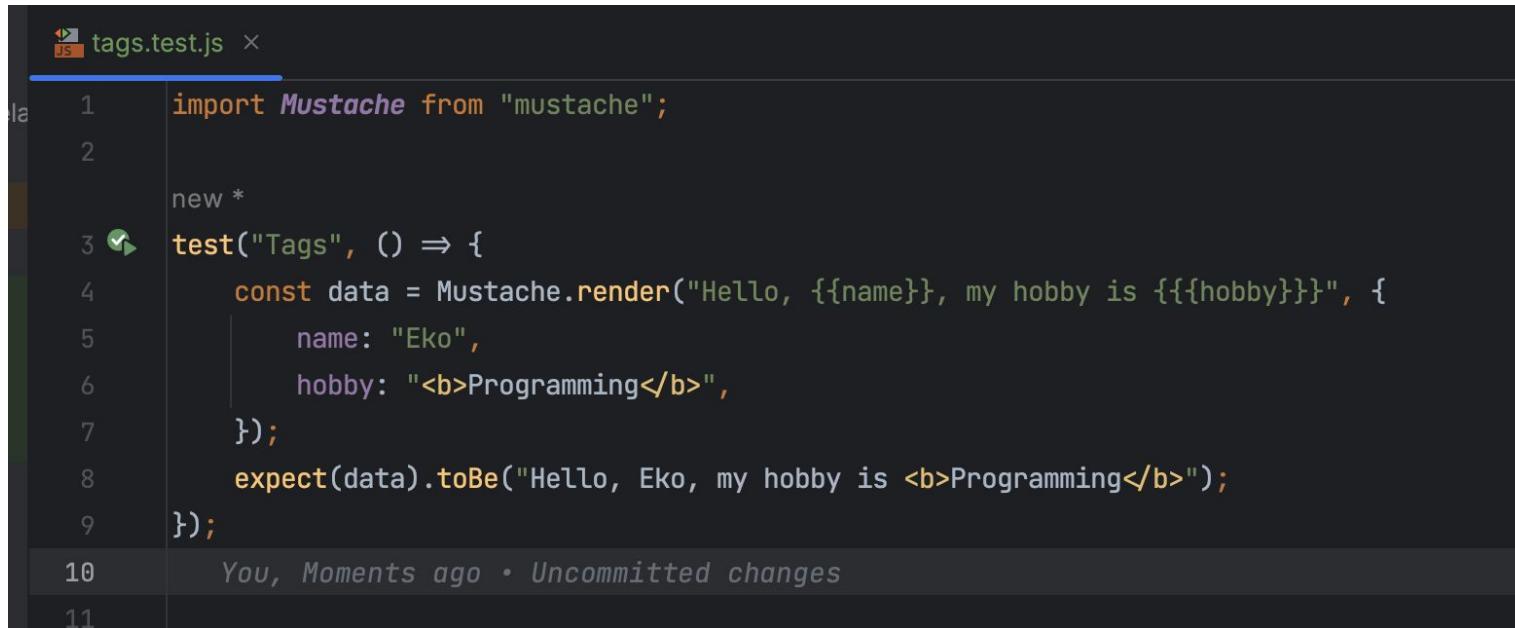
```
1 import Mustache from "mustache";
2
3 new *  
3 ✅ test("Melakukan Compile", () => {
4     Mustache.parse("Hello, {{name}}");
5
6     const data = Mustache.render("Hello, {{name}}", {name: "Eko"});
7     expect(data).toBe("Hello, Eko");
8 });
9 You, Moments ago • Uncommitted changes
10
```

Tags

Tags

- Mustache menggunakan kurung kurawal dua kali sebagai tags, misal {{name}}
- Tag digunakan untuk mengirim data dari luar template, jika data tidak dikirim, maka akan diganti oleh string kosong
- Secara default HTML di dalam data tags akan di escape, jika kita ingin menampilkan HTML, kita bisa gunakan kurung kurawal tiga kali, misal {{{kode}}}

Kode : Tags



A screenshot of a code editor showing a file named `tags.test.js`. The code is a Mocha test for Mustache rendering. It imports Mustache from the mustache module, creates a new test for "Tags", and defines a data object with a name and hobby. The hobby value contains a Mustache tag. An expectation checks if the rendered data matches the expected output.

```
tags.test.js ×
1 import Mustache from "mustache";
2
3 new *
4
5 test("Tags", () => {
6     const data = Mustache.render("Hello, {{name}}, my hobby is {{{hobby}}}", {
7         name: "Eko",
8         hobby: "<b>Programming</b>",
9     });
10    expect(data).toBe("Hello, Eko, my hobby is <b>Programming</b>");
11});
```

You, Moments ago • Uncommitted changes

Nested Object

Nested Object

- Mustache Tags juga bisa digunakan untuk data nested, bahkan array (akan dibahas di materi khusus)
- Untuk mengaksesnya, dalam tags kita bisa gunakan titik, misal {{person.name}}

Kode : Nested Object

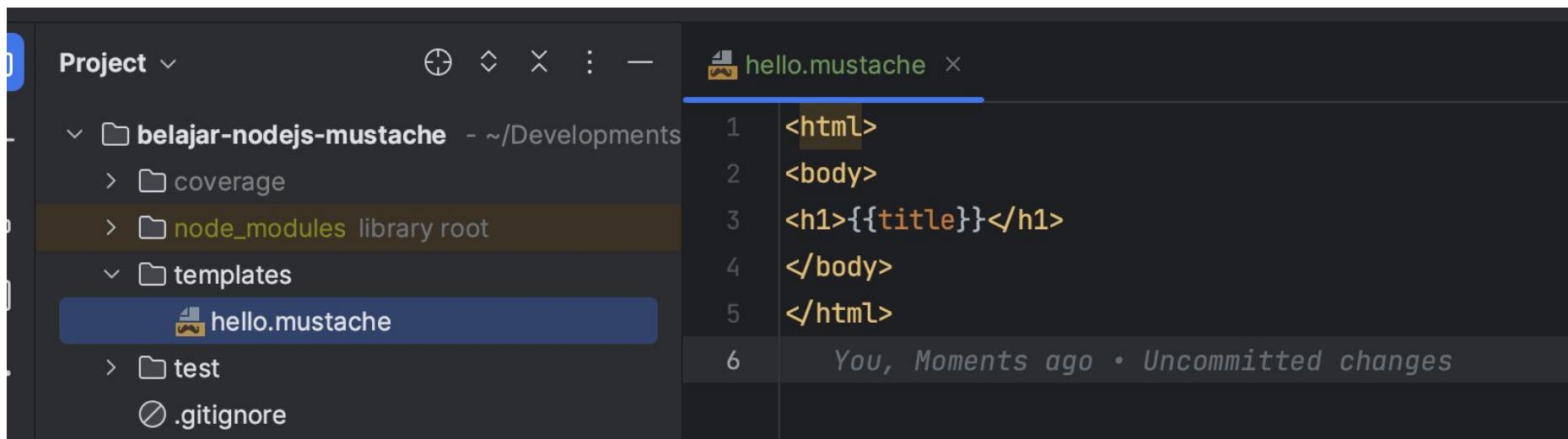
```
11  ✓ < test("Nested Object", () => {
12    <const data = Mustache.render("Hello, {{person.name}}", {
13      <person : {
14        <name: "Eko"
15      }
16    });
17    <expect(data).toBe("Hello, Eko");
18  });
19
```

Mustache File

Mustache File

- Mustache JS sendiri sebenarnya tidak memiliki kemampuan untuk membaca template dari file, jadi kita perlu membaca template dari file secara manual
- Biasanya Mustache menggunakan nama file dengan extension .mustache

Kode : Mustache Template



A screenshot of a code editor interface, likely VS Code, displaying a Mustache template file named `hello.mustache`. The left sidebar shows a project structure with a folder named `belajar-nodejs-mustache` containing subfolders `coverage`, `node_modules` (marked as library root), `templates`, and files `hello.mustache` (selected), `test`, and `.gitignore`. The right pane shows the contents of the `hello.mustache` file:

```
<html>
<body>
<h1>{{title}}</h1>
</body>
</html>
```

At the bottom of the editor, there is a status bar message: *You, Moments ago • Uncommitted changes*.

Kode : Membaca Mustache File

```
tags.test.js ×
1  ✓  import Mustache from "mustache";
2    import fs from "fs/promises"
3
4    new *
5
6    test("Mustache File", async () => {
7      const helloTemplate = await fs.readFile("./templates/hello.mustache")
8        .then(data => data.toString())
9
10     const data = Mustache.render(helloTemplate, {
11       title: "Eko"
12     });
13
14     console.info(data);
15     expect(data).toContain("Eko");
16   });
17 }
```

Sections

Sections

- Mustache adalah template engine yang tidak memiliki logic kompleks
- Tapi kadang-kadang, kita ingin menambahkan logic, misal if else
- Di Mustache, hal ini bisa kita ganti menjadi sections
- Sections di Mustache menggunakan {{#nama}}, dan ditutup dengan {{/nama}}
- Dimana blok di dalam sections tidak akan ditampilkan jika variable di sections tidak ada, atau ada tapi dengan nilai null, undefined, false, 0, NaN, empty string atau empty array



Kode : Mustache File

```
7      <meta http-equiv="X-UA-Compatible" content="ie=edge">
8      <title>Person</title>
9    </head>
10   <body>
11   {{#person}}
12     Hello Person
13   {{/person}}
14   </body>
15   </html>
```

| You, Moments ago • Uncommitted changes



Kode : Sections Not Show

```
new *  
6 ►✓ test("Sections Not Show", async () => {  
7     const helloTemplate = await fs.readFile("./templates/person.mustache")  
8         .then(data => data.toString());  
9  
0     const data = Mustache.render(helloTemplate, {});  
1  
2     console.info(data);  
3     expect(data).not.toContain("Hello Person");  
4});  
5
```

Kode : Sections Show

```
✓ test("Sections Show", async () => {
  const helloTemplate = await fs.readFile("./templates/person.mustache")
    .then(data => data.toString());

  const data = Mustache.render(helloTemplate, {
    person: {
      name: "Eko"
    }
  });

  console.info(data);
  expect(data).toContain("Hello Person");
})
```

Sections Data

Sections Data

- Saat menggunakan sections, kita juga bisa mengakses data yang terdapat di sections secara otomatis, tanpa harus menyebutkan sections nya lagi
- Misal pada kasus sebelumnya, kita bisa mengakses {{name}} tanpa harus menyebutkan {{person.name}}



Kode : Mustache File

```
8      <title>Person</title>
9  </head>
10 <body>
11 {{#person}}
12     Hello Person {{name}}!
13 {{/person}}
14 </body>
15 </html>
16 You, Moments ago • Uncommitted changes
```



Kode : Sections Data

```
0  test("Sections Data", async () => {
1      const helloTemplate = await fs.readFile("./templates/person.mustache")
2          .then(data => data.toString());
3
4      const data = Mustache.render(helloTemplate, {
5          person: {
6              name: "Eko"
7          }
8      });
9
10     console.info(data);
11     expect(data).toContain("Hello Person Eko!");
12 });


```

Inverted Sections

Inverted Sections

- Saat menggunakan logika if, kadang kita ingin menambahkan kondisi else
- Di Mustache, kita bisa menggunakan inverted sections, dimana blok sections akan dieksekusi jika nilai sections adalah null, undefined, false, falsy atau empty list
- Untuk menggunakan Inverted Sections, kita bisa gunakan {{^nama}} dan ditutup dengan {{/nama}}



Kode : Mustache File

```
9  </head>
10 <body>
11 {{#person}}
12   Hello Person {{name}}!
13 {{/person}}
14 {{^person}}
15   Hello Guest
16 {{/person}}
17 </body>
18 </html>
```

Kode : Inverted Sections

```
new *  
54 ✓ < test("Inverted Sections", async () => {  
55     const helloTemplate = await fs.readFile("./templates/person.mustache")  
56     .then(data => data.toString());  
57  
58     const data = Mustache.render(helloTemplate, {});  
59  
60     console.info(data);  
61     expect(data).toContain("Hello Guest");  
62 });  
63 You, Moments ago • Uncommitted changes
```

List

List

- Saat membuat template, kadang kita ingin menampilkan data yang bentuknya adalah List atau Array
- Mustache juga bisa menggunakan sections untuk menampilkan itu
- Untuk menampilkan tiap data, kita bisa gunakan titik di dalam sections nya, misal {{.}}
- Ingat, sama seperti sections, jika data list tidak ada, maka tidak akan menampilkan apapun



Kode : Mustache File

```
9  </head>
10 <body>
11 <h1>Hobbies</h1>
12 <ul>
13   {{#hobbies}}
14     <li>{{.}}</li>
15   {{/hobbies}}
16 </ul>
17 </body>
18 </html>
```

Kode : List

```
5 test("List", async () => {
6     const helloTemplate = await fs.readFile("./templates/hobbies.mustache")
7     .then(data => data.toString());
8
9     const data = Mustache.render(helloTemplate, {
10         hobbies: ["Coding", "Gaming", "Reading"]
11     });
12
13     console.info(data);
14     expect(data).toContain("Coding");
15});
```

List Object

List Object

- Kadang, kita memiliki data kumpulan object di dalam List atau Array
- Itu juga bisa kita akses menggunakan sections, caranya mirip dengan menggunakan Sections Object



Kode : Mustache File

```
L0 <body>
L1 <h1>Students</h1>
L2 <table>
L3   {{#students}}
L4     <tr>
L5       <td>{{name}}</td>
L6       <td>{{value}}</td>
L7     </tr>
L8   {{/students}}
L9 </table>
L0 </body>
```



Kode : List Object

```
✓ test("List Object", async () => {
  const helloTemplate = await fs.readFile("./templates/students.mustache")
    .then(data => data.toString());

  const data = Mustache.render(helloTemplate, {
    students : [
      { name: "Eko", value: 100},
      { name: "Budi", value: 100}
    ]
  });

  console.info(data);
  expect(data).toContain("Eko");
});
```

Functions

Functions

- Dalam tags, selain data, kita juga bisa mengakses function
- Caranya cukup mudah, kita hanya perlu membuat function yang mengembalikan function dengan parameter text, dan render
- Parameter render adalah function Mustache untuk melakukan render, dan sebelum dikembalikan, kita bisa memanipulasi hasilnya
- Untuk menggunakan function, caranya sama dengan menggunakan sections, cukup gunakan {{#function}} dan ditutup dengan {{/function}}

Kode : Functions

```
new "test"
92 ►▶ test("Functions", async () => {
93   const parameter = {
94     name: "Eko",
95     upper: () => {
96       return (text, render) => {
97         return render(text).toUpperCase();
98       }
99     }
00   }
01 }
```

Kode : Menggunakan Functions

```
test("Functions", async () => {
  const parameter = {
    name: "Eko",
    upper: () => {
      return (text, render) => {
        return render(text).toUpperCase();
      }
    }
  }

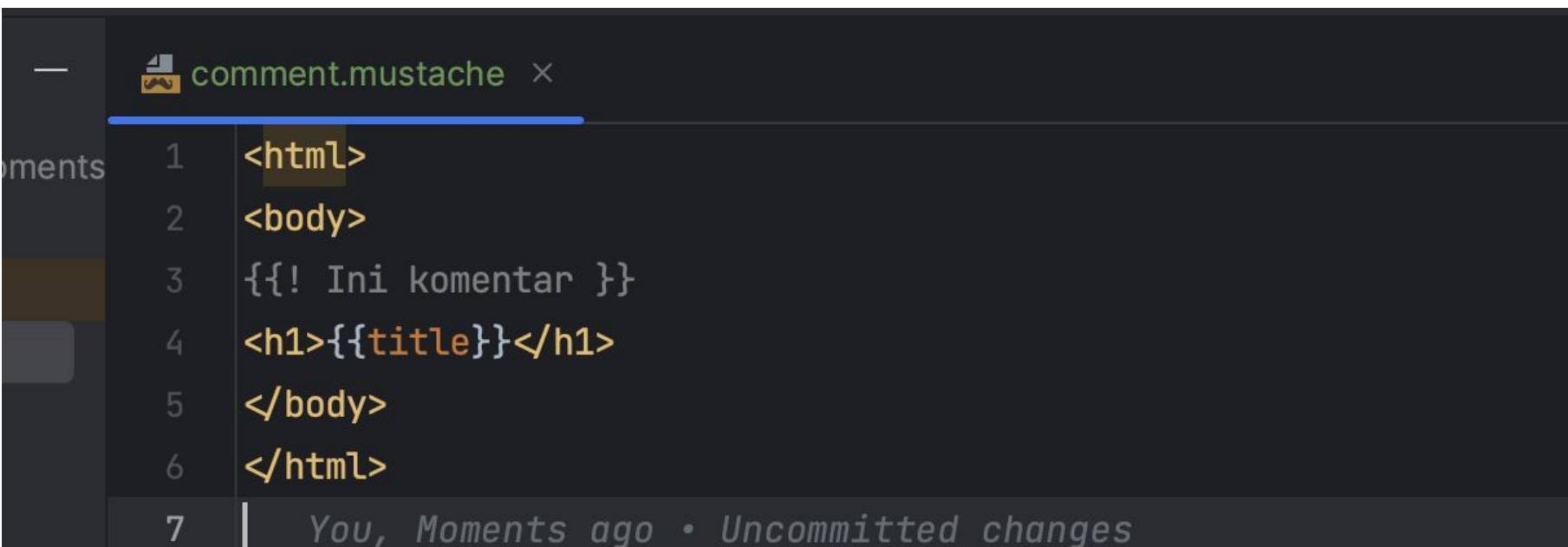
  const data = Mustache.render("Hello {{#upper}}{{name}}{{/upper}}", parameter);
  expect(data).toBe("Hello EKO")
});
```

Comment

Comment

- Kadang kita ingin menambahkan komentar di file Mustache
- Untuk menambahkan komentar, kita bisa menggunakan {{! isi komentar }}

Kode : Mustache File



A screenshot of a code editor interface, likely VS Code, showing a file named "comment.mustache". The file contains a Mustache template for generating HTML. The code is as follows:

```
1 <html>
2 <body>
3 {{! Ini komentar }}
4 <h1>{{title}}</h1>
5 </body>
6 </html>
```

The file is currently uncommitted, as indicated by the status bar at the bottom.



Kode : Comment

```
▶ test("Comment", async () => {
    const helloTemplate = await fs.readFile("./templates/comment.mustache")
        .then(data => data.toString());

    const data = Mustache.render(helloTemplate, {title: "Eko"});

    console.info(data);
    expect(data).toContain("Eko");
    expect(data).not.toContain("Komentar");
});
```

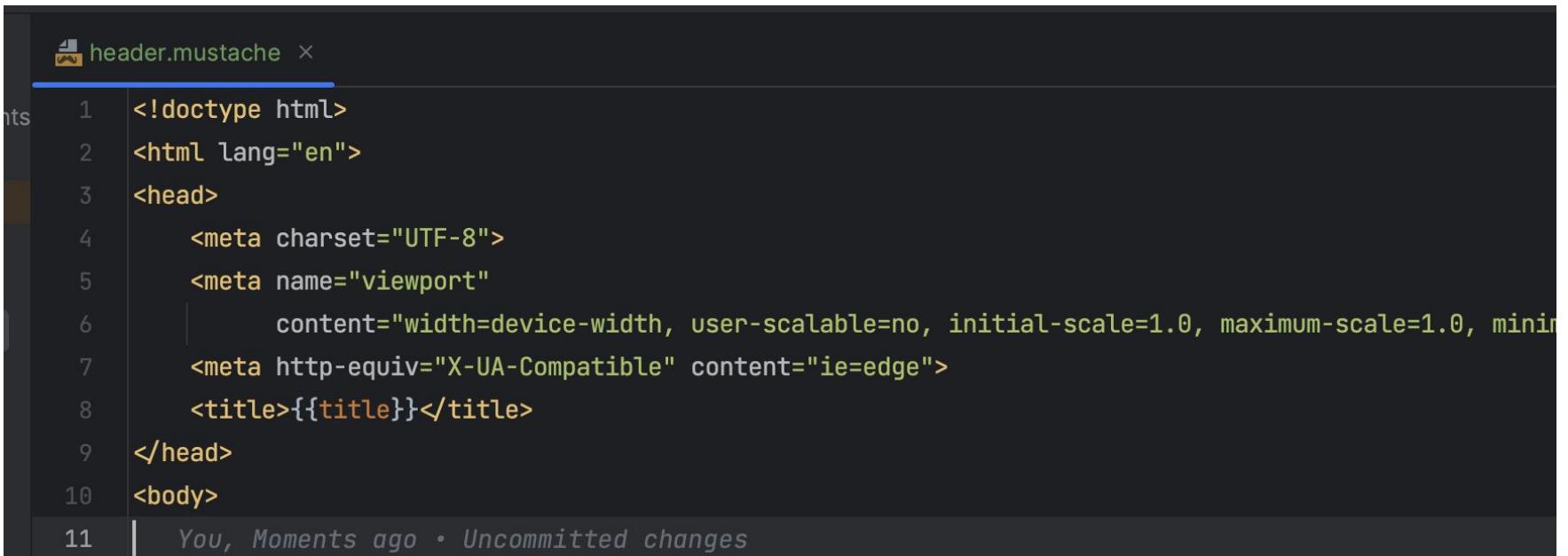
Partials

Partials

- Saat kita membuat halaman web menggunakan Mustache, kadang kita ingin membagi template menjadi beberapa bagian
- Misal ada bagian header, content dan footer
- Untungnya, Mustache mendukung hal tersebut
- Kita bisa menggunakan perintah {{> namaPartial}}
- Pada function render, terdapat parameter ketika berisikan object partials yang bisa kita gunakan untuk menambahkan data template lainnya



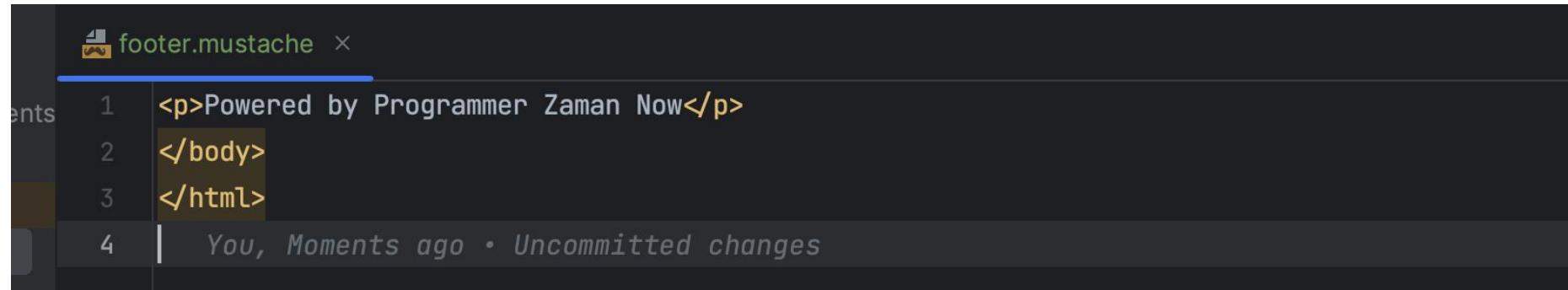
Kode : Mustache Header



A screenshot of a code editor showing a file named "header.mustache". The code is a Mustache template for an HTML header. It includes meta tags for charset, viewport, and http-equiv, and a title tag with a placeholder {{title}}. The code is numbered from 1 to 11.

```
header.mustache
1 <!doctype html>
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">
5     <meta name="viewport"
6         content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0, minin
7     <meta http-equiv="X-UA-Compatible" content="ie=edge">
8     <title>{{title}}</title>
9 </head>
10 <body>
11 | You, Moments ago • Uncommitted changes
```

Kode : Mustache Footer



A screenshot of a code editor showing a file named "footer.mustache". The file contains the following HTML code:

```
1 <p>Powered by Programmer Zaman Now</p>
2 </body>
3 </html>
4 | You, Moments ago • Uncommitted changes
```

The code editor has a dark theme with syntax highlighting. The file icon is a mustache icon. The status bar on the left shows "ents". The bottom right corner of the code area has a watermark-like text: "You, Moments ago • Uncommitted changes".

Kode : Mustache Content

```
content.mustache ×  
1 {{> header}}  
2  
3 <h1>{{title}}</h1>  
4 <p>{{content}}</p>  
5  
6 {{> footer}}  
7 You, Moments ago • Uncommitted changes
```

Kode : Partials

```
✓ test("Partials", async () => {
    const headerTemplate = await fs.readFile("./templates/header.mustache").then(data => data.toString());
    const footerTemplate = await fs.readFile("./templates/footer.mustache").then(data => data.toString());
    const contentTemplate = await fs.readFile("./templates/content.mustache").then(data => data.toString());

    const data = Mustache.render(contentTemplate, {
        title: "Belajar Partials",
        content: "Eko Kurniawan Khannedy"
    }, {
        header: headerTemplate,
        footer: footerTemplate
    });

    console.info(data);
    expect(data).toContain("Belajar Partials");
    expect(data).toContain("Eko Kurniawan Khannedy");
    expect(data).toContain("Programmer Zaman Now");
});
```

Materi Selanjutnya

Materi Selanjutnya

- NodeJS Database