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
- Facebook : fb.com/ProgrammerZamanNow
- Instagram : instagram.com/programmerzamannow
- Youtube : youtube.com/c/ProgrammerZamanNow
- Telegram Channel : t.me/ProgrammerZamanNow
- Tiktok : https://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

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

```
  "author": "Eko Kurniawan Khannedy",
  "license": "ISC",
  "dependencies": {
    "mustache": "^4.2.0"
  },
  "devDependencies": {
    "@babel/plugin-transform-runtime": "^7.17.12",
    "@babel/preset-env": "^7.17.12",
    "babel-jest": "^28.1.0",
    "jest": "^28.1.0",
    "@types/jest": "^28.1.0",
    "@types/mustache": "^4.2.0"
  }
}
```

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

mustache.test.js

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

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
   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  });   You, Moments ago • Uncommitted changes
9
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

tags.test.js

```
import Mustache from "mustache";

new *
test("Tags", () => {
    const data = Mustache.render("Hello, {{name}}, my hobby is {{{hobby}}}", {
        name: "Eko",
        hobby: "<b>Programming</b>",
    });
    expect(data).toBe("Hello, Eko, my hobby is <b>Programming</b>");
});
   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

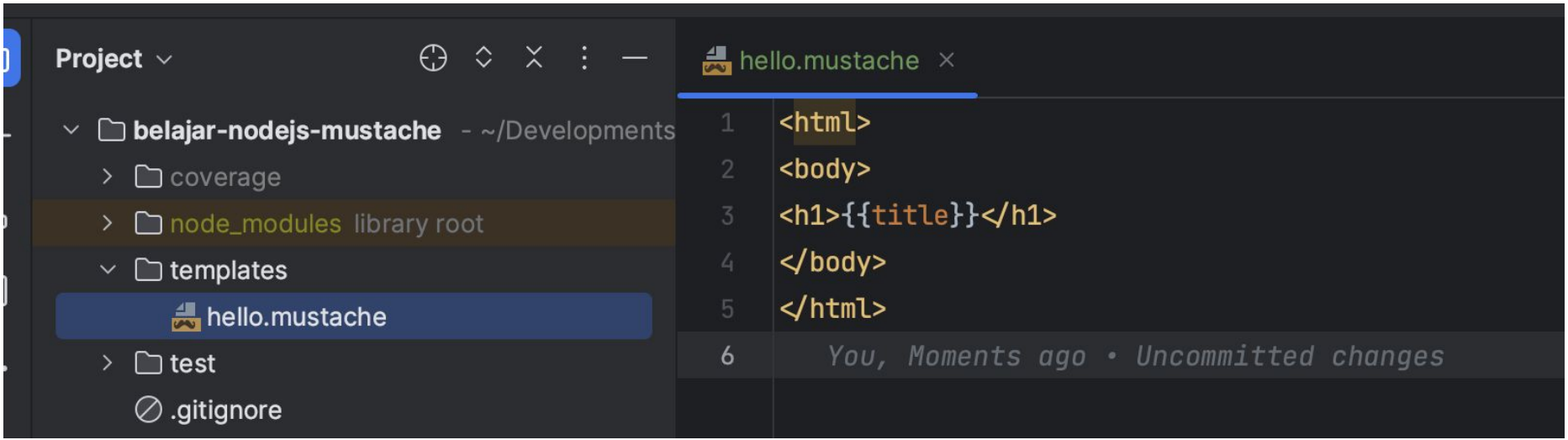
```
11  test("Nested Object", () => {
12      const data = Mustache.render("Hello, {{person.name}}", {
13          person : {
14              name: "Eko"
15          }
16      });
17      expect(data).toBe("Hello, Eko");
18  });
19
```

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



Kode : Membaca Mustache File

tags.test.js

```
import Mustache from "mustache";
import fs from "fs/promises"

new *
test("Mustache File", async () => {
    const helloTemplate = await fs.readFile("./templates/hello.mustache")
        .then(data => data.toString());

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

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

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>
16    You, Moments ago • Uncommitted changes
```

Kode : Sections Not Show

```
new *
test("Sections Not Show", async () => {
    const helloTemplate = await fs.readFile("./templates/person.mustache")
        .then(data => data.toString());

    const data = Mustache.render(helloTemplate, {});

    console.info(data);
    expect(data).not.toContain("Hello Person");
});
```

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

```
test("Sections Data", 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 Eko!");
});
```

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

```
</head>
<body>
{{#person}}
    Hello Person {{name}}!
{{/person}}
{{^person}}
    Hello Guest
{{/person}}
</body>
</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

```
</head>
<body>
<h1>Hobbies</h1>
<ul>
    {{#hobbies}}
        <li>{{.}}</li>
    {{/hobbies}}
</ul>
</body>
</html>
```

Kode : List

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

    const data = Mustache.render(helloTemplate, {
        hobbies: ["Coding", "Gaming", "Reading"]
    });

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

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

```
<body>
<h1>Students</h1>
<table>
    {{#students}}
        <tr>
            <td>{{name}}</td>
            <td>{{value}}</td>
        </tr>
    {{/students}}
</table>
</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 *
92  test("Functions", async () => {
93      const parameter = {
94          name: "Eko",
95          upper: () => {
96              return (text, render) => {
97                  return render(text).toUpperCase();
98              }
99          }
100     }
```

Kode : Menggunakan Functions

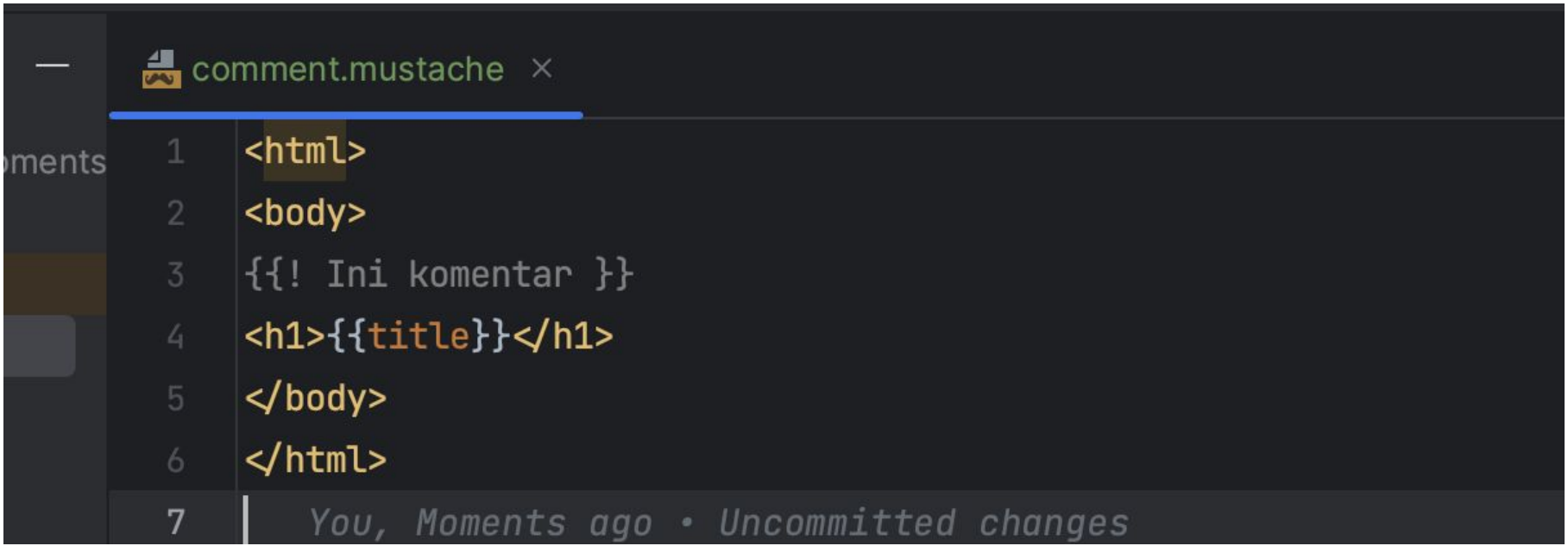
```
test("Functions", async () => {
    const parameter = {
        name: "Eko",
        upper: () => {
            return (text, render) => {
                return render(text).toUpperCase();
            }
        }
    }

    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



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

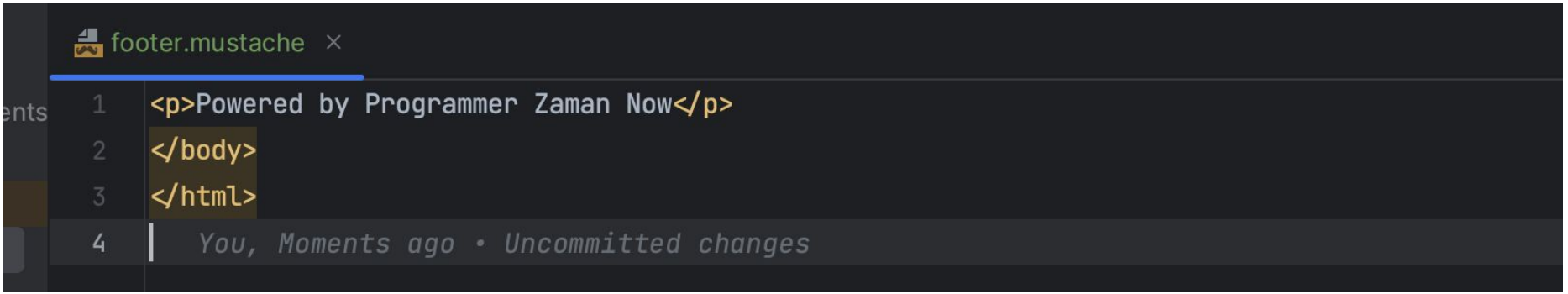
- Saat kita membuat halaman web menggunakan Mustache, kadang kita ingin membagi template menjadi beberapa bagian
- Misal ada bagian header, content dan footer
- Untungnya, Mustache mendukung hal tersebut
- Kita bisa menggunakan perintah {{> namaPartial}}
- Pada function render, terdapat parameter ketika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

Kode : Mustache Header

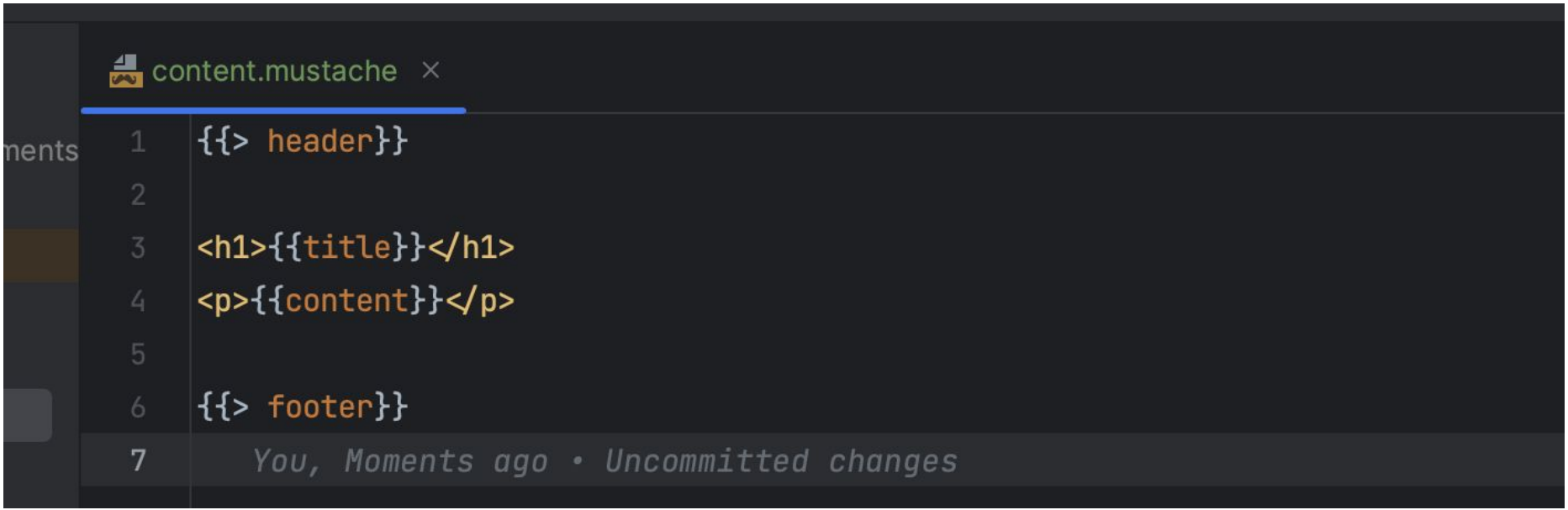
header.mustache

```
<!doctype html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
          content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0, minim
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>{{title}}</title>
</head>
<body>
  You, Moments ago • Uncommitted changes
```

Kode : Mustache Footer



Kode : Mustache Content



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