Templates

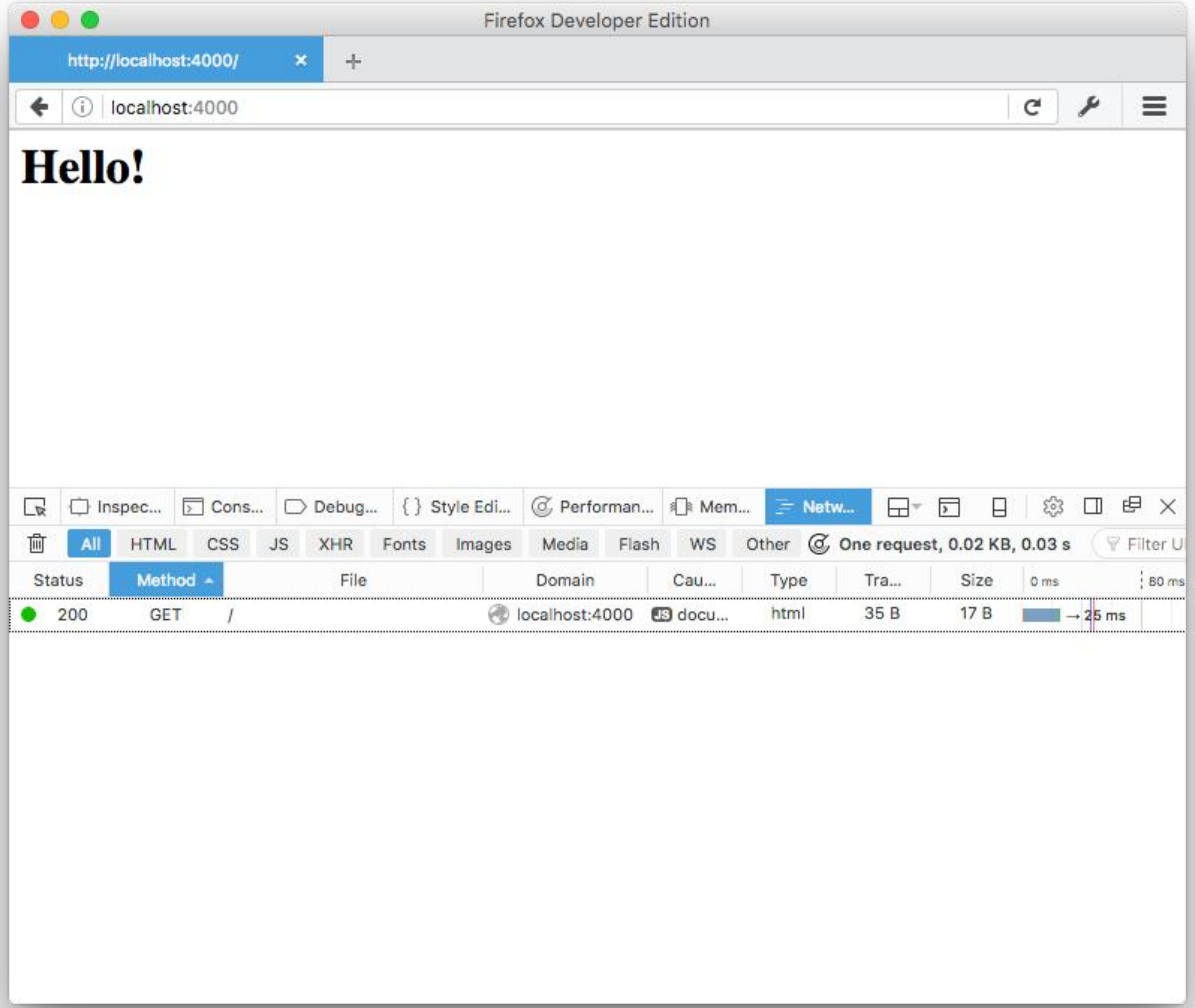
response.send

- In order to render web pages we could pass html content.

- This would become very unwieldy and unmaintainable.

start.js

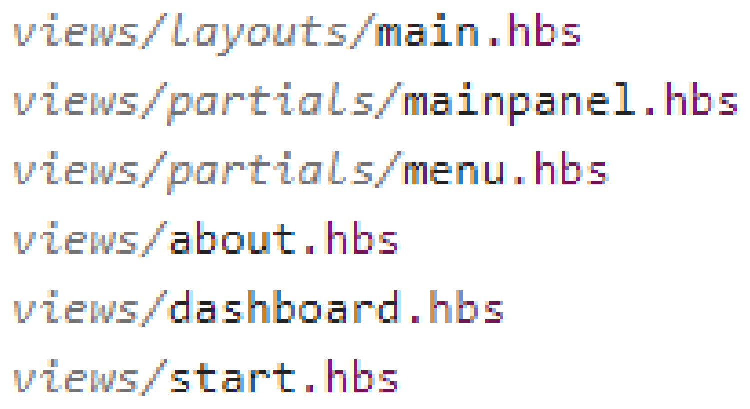
```
const start = {
  index(request, response) {
    logger.info('start rendering');
    response.send('<h1> Hello </h1>');
  },
};
```


Firefox Developer Edition
http://localhost:4000/
localhost:4000
Inspec... Cons... Debug... Style Edi... Performan... Mem... Netw...
All HTML CSS JS XHR Fonts Images Media Flash WS Other
One request, 0.02 KB, 0.03 s

Hello!

Status	Method	File	Domain	Cau...	Type	Tra...	Size	0 ms
200	GET	/	localhost:4000	docu...	html	35 B	17 B	→ 25 ms

Front-end



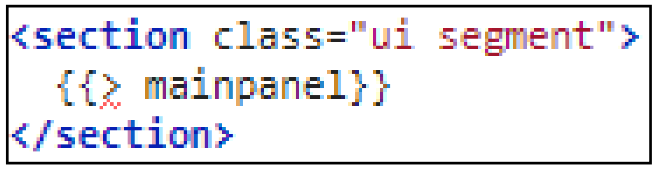
```
views/layouts/main.hbs
views/partials/mainpanel.hbs
views/partials/menu.hbs
views/about.hbs
views/dashboard.hbs
views/start.hbs
```

- All written in HTML + handlebars.
- Handlebars: Templating language.
- It supports:
  - **Layouts**
  - **Partials**
  - **Views**

Partials & Layouts

- Partials & Layouts play a prominent role in enabling DRY (Don’t Repeat Yourself) principles.
  - Layouts: Reusable Page Structure.
  - Partials: Reusable templates.

```
<!DOCTYPE html>
<html>
 <head>
   <meta charset="utf-8">
   <title> {{title}} </title>
   <meta charset="UTF-8">
   <script type="text/javascript" src="https://cdnjs.cloudflare.com/a
   <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs
   <script type="text/javascript" src="https://cdnjs.cloudflare.com/a
  <link rel="stylesheet" type="text/css" href="/stylesheets/style.css
 </head>
 <body>
   <section class="ui container">
     {{{body}}}
   </section>
 </body>
</html>
```



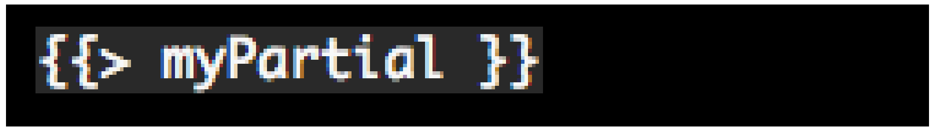
```
<section class="ui segment">
  {{> mainpanel}}
</section>
```

```
<h1 class="ui header">
  Title for Dashboard Panel
</h1>
<p>
  To be replaced with content...
</p>
```

Partials

- Handlebars partials allow for code reuse by creating shared templates.

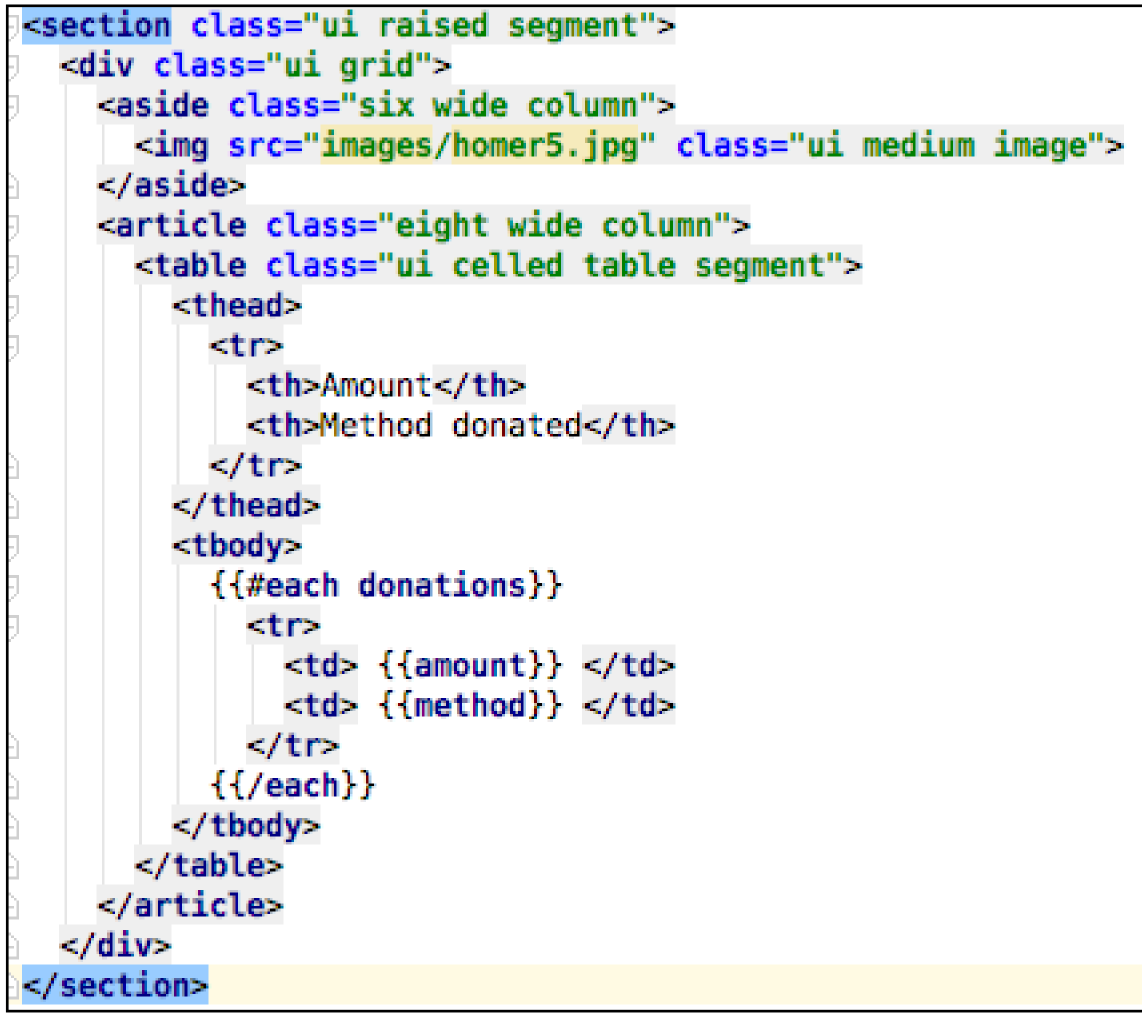
- Calling the partial is done through the partial call syntax:



```
{{> myPartial }}
```

- Will render the partial named myPartial. When the partial executes, it will be run under the current execution context.

myPartial.hbs



```
<section class="ui raised segment">
  <div class="ui grid">
    <aside class="six wide column">
      <img src="images/homer5.jpg" class="ui medium image">
    </aside>
    <article class="eight wide column">
      <table class="ui celled table segment">
        <thead>
          <tr>
            <th>Amount</th>
            <th>Method donated</th>
          </tr>
        </thead>
        <tbody>
          {{#each donations}}
            <tr>
              <td> {{amount}} </td>
              <td> {{method}} </td>
            </tr>
          {{/each}}
        </tbody>
      </table>
    </article>
  </div>
</section>
```

Layout

- All views will be based on structure laid down in **main.hbs.**
- Includes Semantic-UI CSS library.
- View content will be inserted into {{{body}}}

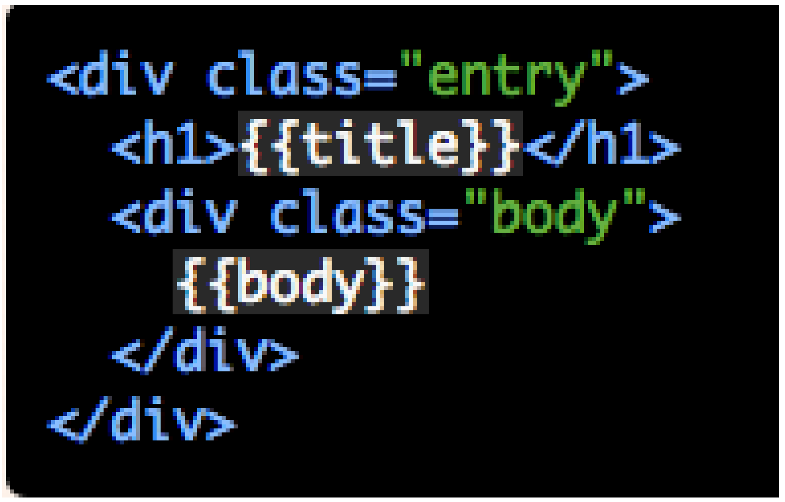
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title> {{title}} </title>
    <meta charset="UTF-8">
    <script type="text/javascript" src="https://cdnjs.cloudflare.com/ajax/libs/jque
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/semantic-ui
    <script type="text/javascript" src="https://cdnjs.cloudflare.com/ajax/libs/sema
    <link rel="stylesheet" type="text/css" href="/stylesheets/style.css">
  </head>
  <body>
    <section class="ui container">
      {{{body}}}
    </section>
  </body>
</html>
```

Template Expressions

- In addition to layouts + partials, templating also support **Template Expressions.**

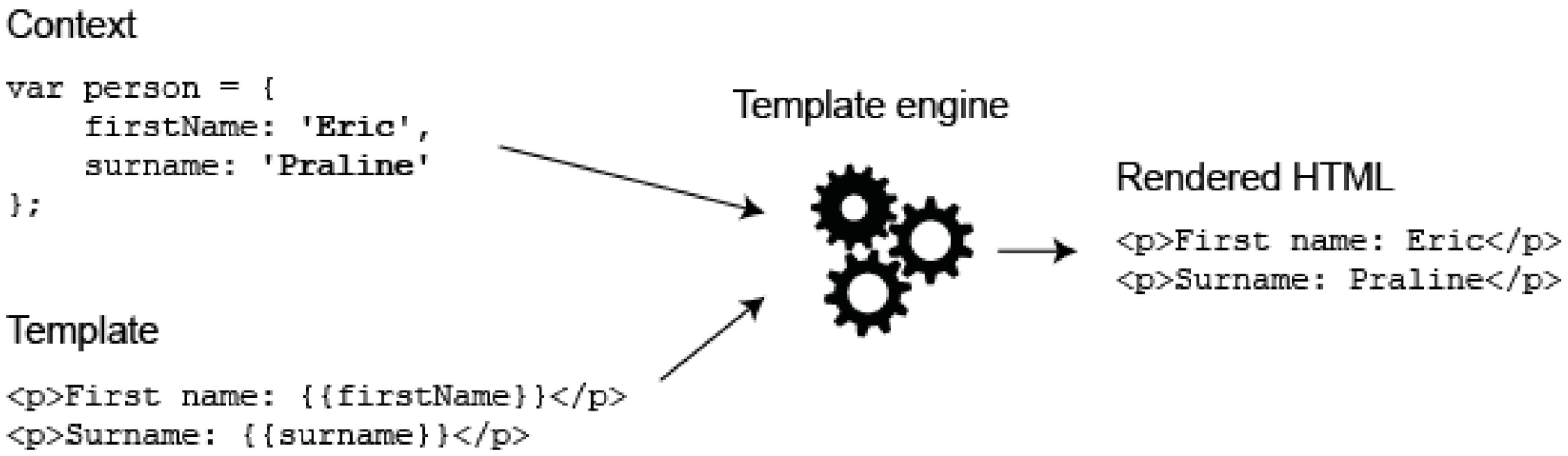
- These expressions enable external information to be incorporated into a page.

- This information will be delivered via JavaScript Objects.



```
<div class="entry">
  <h1>{{title}}</h1>
  <div class="body">
    {{body}}
  </div>
</div>
```

Templating Engine



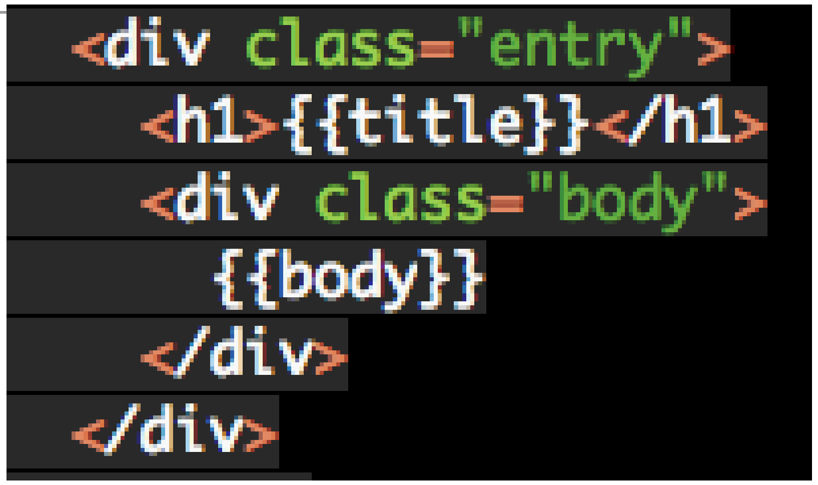
```
var person = {
    firstName: 'Eric',
    surname: 'Praline'
};
```

```
<p>First name: {{firstName}}</p>
<p>Surname: {{surname}}</p>
```

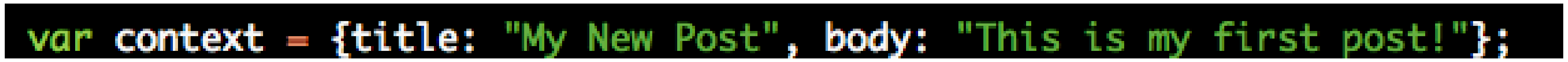
```
<p>First name: Eric</p>
<p>Surname: Praline</p>
```

Template Expressions

- A handlebars expression is a {{, some contents, followed by a }}



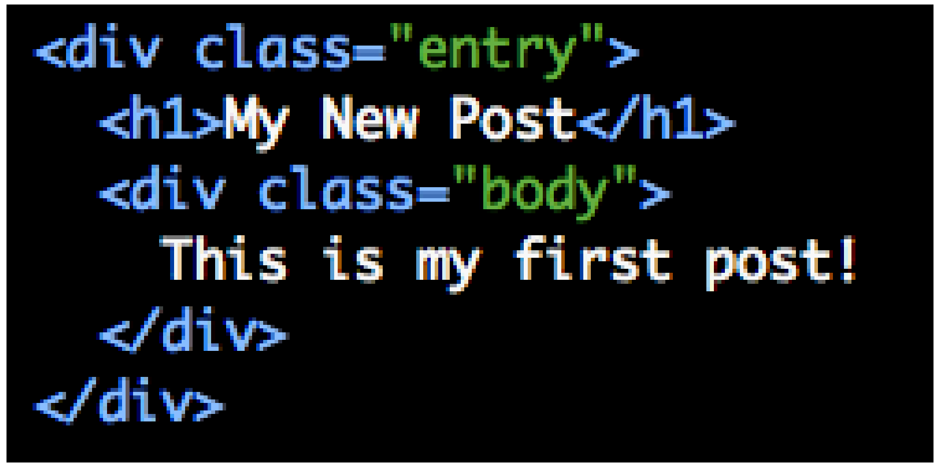
```
<div class="entry">
  <h1>{{title}}</h1>
  <div class="body">
    {{body}}
  </div>
</div>
```



```
var context = {title: "My New Post", body: "This is my first post!"};
```

- In JavaScript, create an object literal with matching properties.

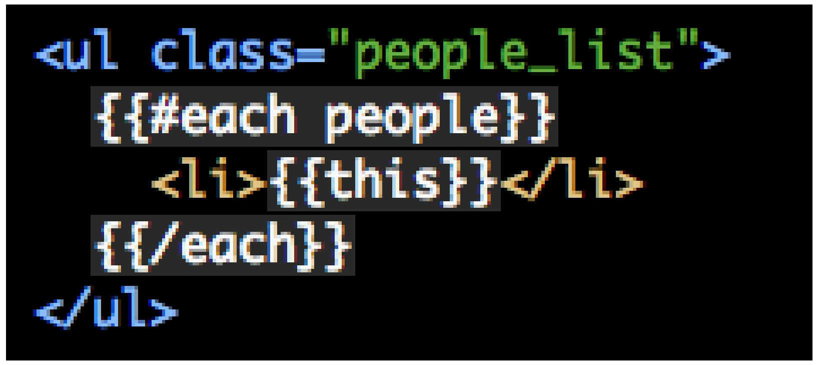
- When rendered, the properties replace the handlebars expressions.



```
<div class="entry">
  <h1>My New Post</h1>
  <div class="body">
    This is my first post!
  </div>
</div>
```

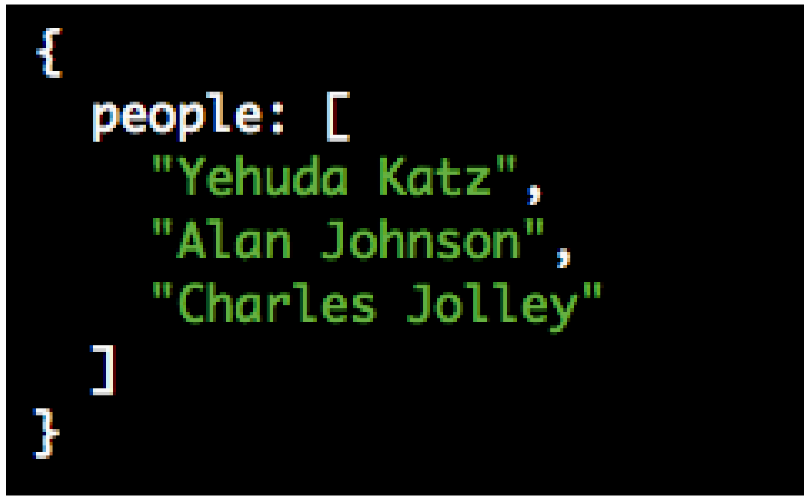
each helper

You can iterate over a list using the built-in **each** helper. Inside the block, you can use *this* to reference the element being iterated over.



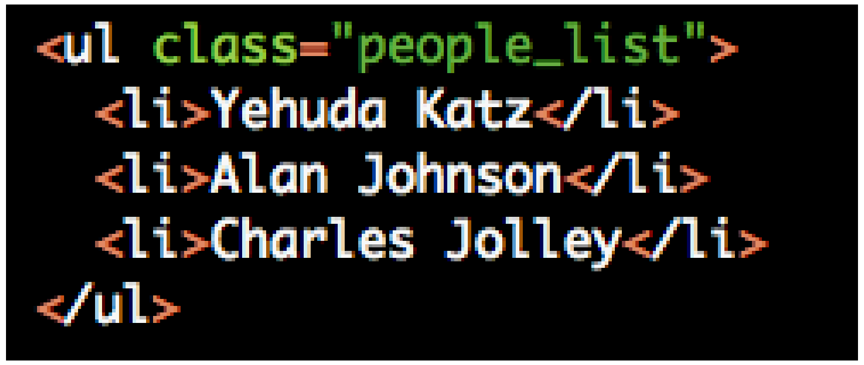
```
<ul class="people_list">
  {{#each people}}
    <li>{{this}}</li>
  {{/each}}
</ul>
```

when used with this context:



```
{
  people: [
    "Yehuda Katz",
    "Alan Johnson",
    "Charles Jolley"
  ]
}
```

will result in:

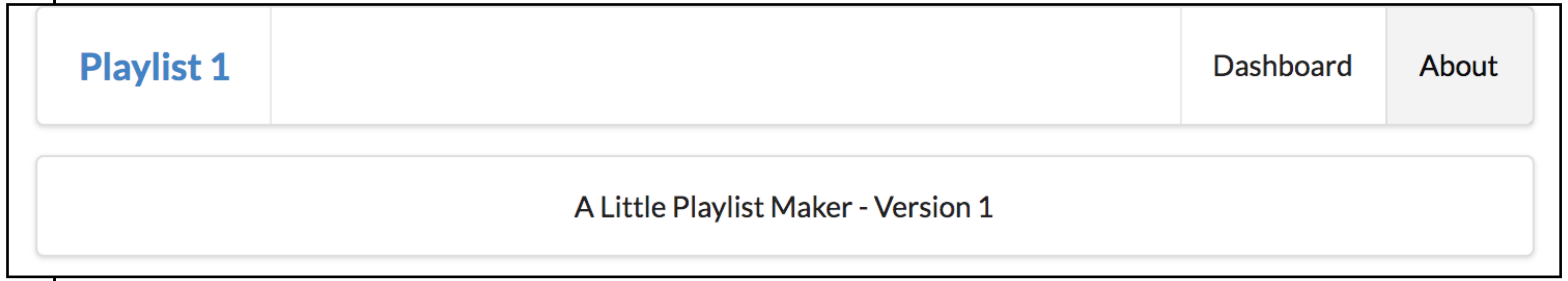


```
<ul class="people_list">
  <li>Yehuda Katz</li>
  <li>Alan Johnson</li>
  <li>Charles Jolley</li>
</ul>
```

About Controller -> About View

about.js

```
'use strict';

const logger = require('../utils/logger');

const about = {
  index(request, response) {
    logger.info('about rendering');
    const viewData = {
      title: 'About Playlist 1',
    };
    response.render('about', viewData);
  },
};

module.exports = ab
```



A Little Playlist Maker - Version 1

about.hbs

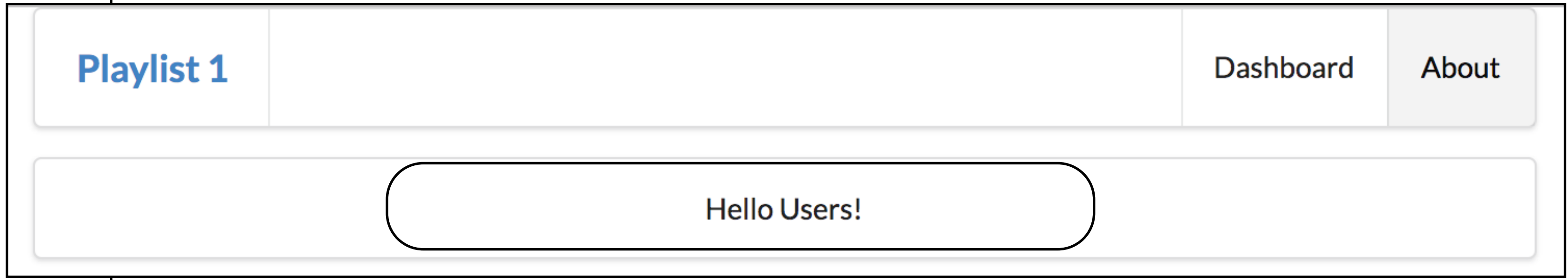
```
{{> menu id="about"}}

<section class="ui center aligned middle aligned segment">
  <p>
    A Little Playlist Maker - Version 1
  </p>
</sect
```

- **response.render** locates the named template and sends it to the browser.
- It also passes the **viewData** object to the view.

About Controller -> About View

about.js

```
'use strict';

const logger = require('../utils/logger');

const about = {
  index(request, response) {
    logger.info('about rendering');
    const viewData = {
      title: 'About Playlist 1',
      greeting: 'Hello Users!',
    };
    response.render('about', viewData);
  },
};

module.exports = ab
```



Hello Users!

about.hbs

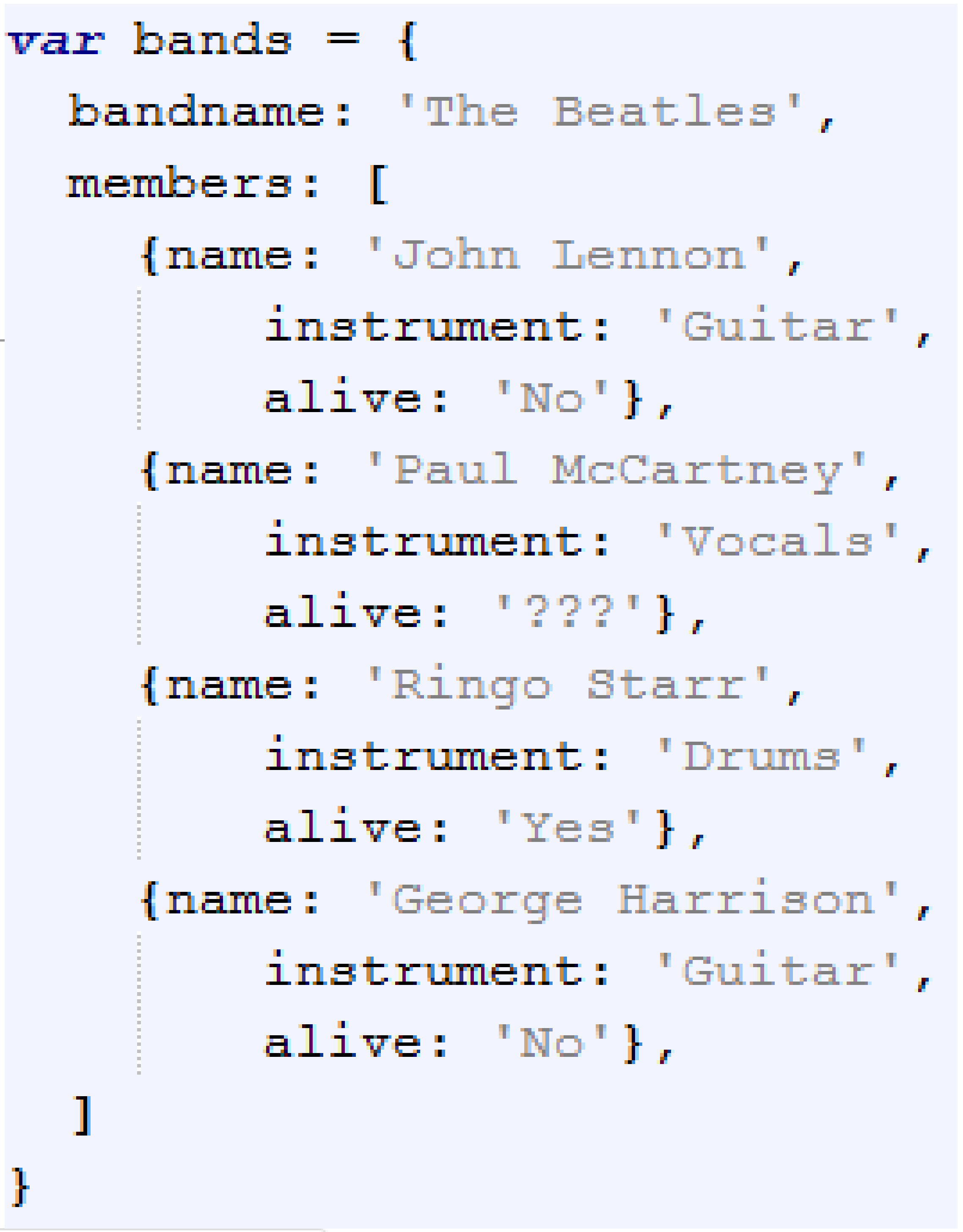
```
{{> menu id="about"}}

<section class="ui center aligned middle aligned segment">
  <p>
   {{greeting}}
  </p>
</section>
```

- We can pass simple and complex data to the views.

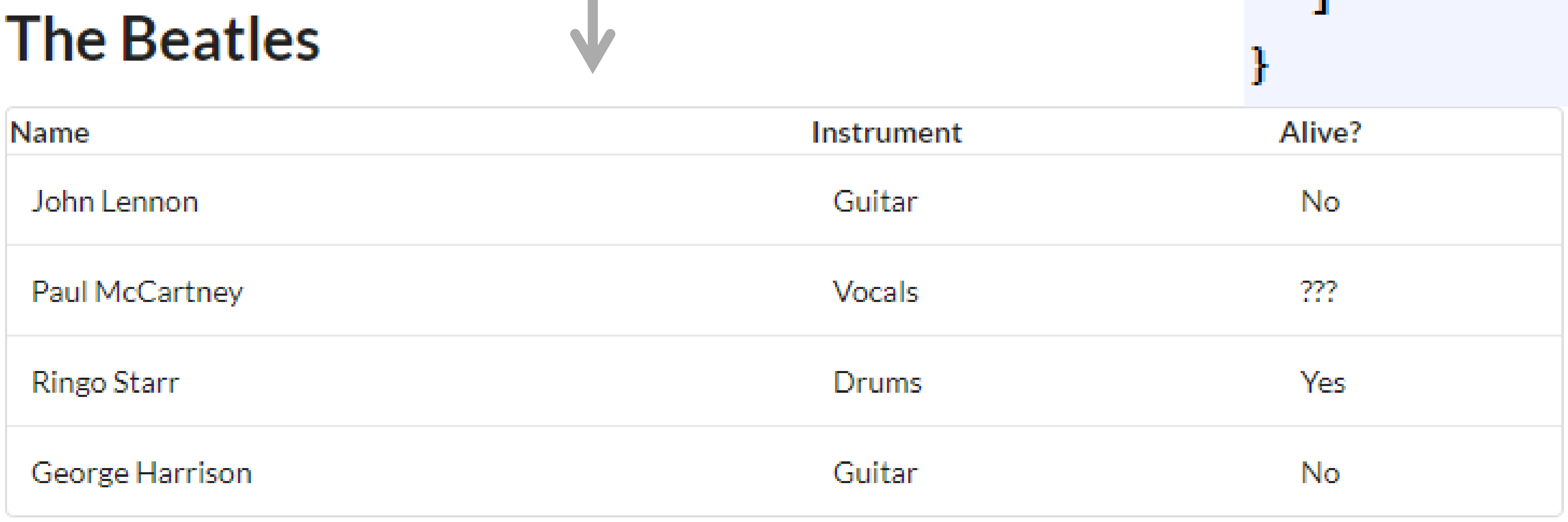
- {{greeting}} replaced with the value in the viewData object called ‘greeting’.

Class exercise

- Given this context
- Write the code for a template that will output the heading and table shown below:



```
var bands = {
  bandname: 'The Beatles',
  members: [
     {name: 'John Lennon',
         instrument: 'Guitar',
         alive: 'No'},
     {name: 'Paul McCartney',
         instrument: 'Vocals',
         alive: '???'},
     {name: 'Ringo Starr',
         instrument: 'Drums',
         alive: 'Yes'},
     {name: 'George Harrison',
         instrument: 'Guitar',
         alive: 'No'},
  ]
}
```



The Beatles

Name	Instrument	Alive?
John Lennon	Guitar	No
Paul McCartney	Vocals	???
Ringo Starr	Drums	Yes
George Harrison	Guitar	No