

JavaScript (шаблонизация)

Mustache.js — шаблонизатор для JavaScript

Библиотека

<https://github.com/janl/mustache.js>

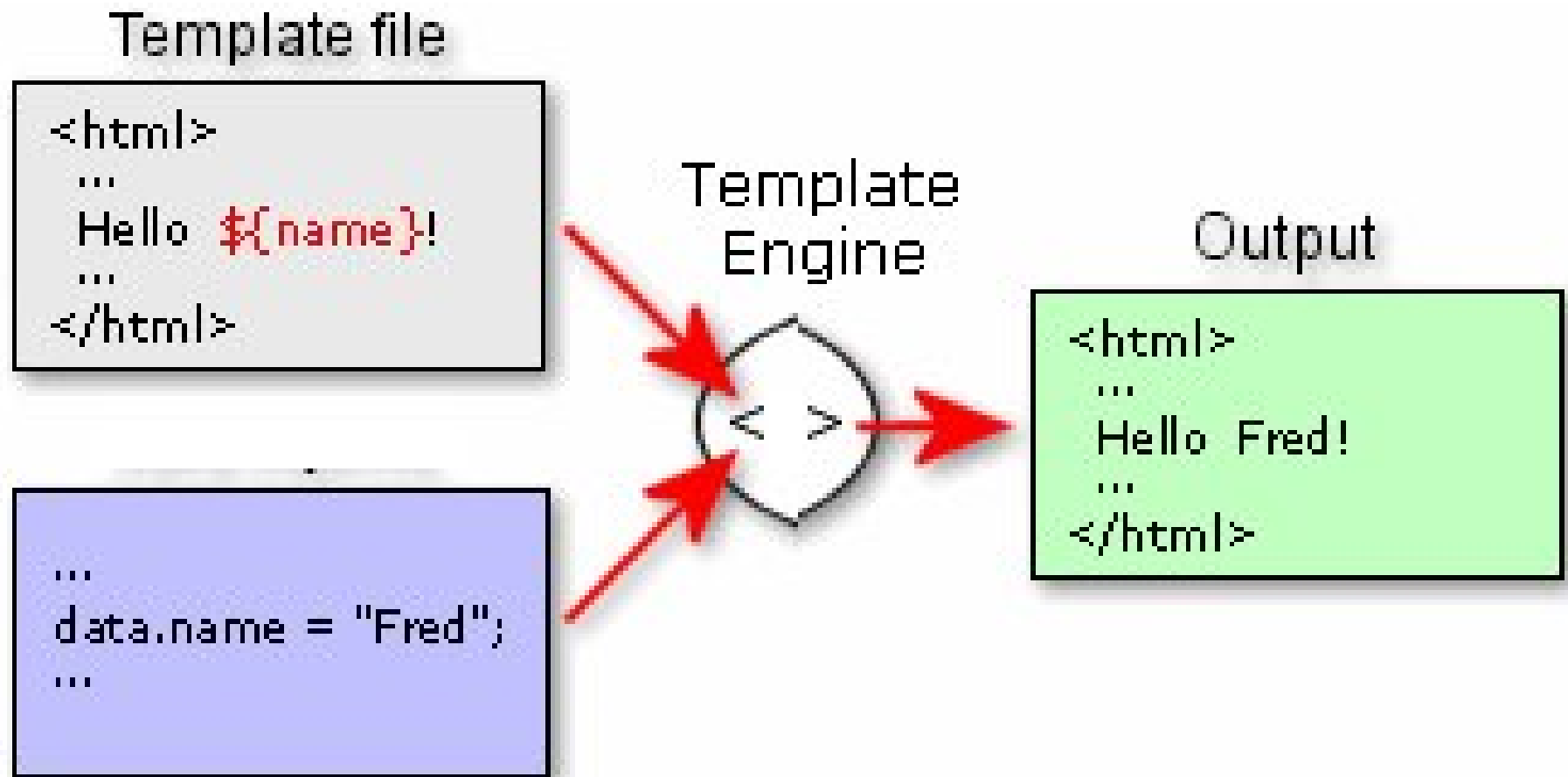
Документация

<https://github.com/janl/mustache.js#usage>

Подключение mustache.js

```
<html>  
  <script src="mustache.js"></script>  
  <script type="text/javascript">  
    // Код из презентации  
  </script>  
</html>
```

Шаблонизация



Шаблонизация

Шаблонизация — подстановка конкретных значений в заданный шаблон. Основные теги для замены в шаблоне выглядят так `{{ }}`. В них содержится название поля json объекта значение которого будет подставляться в место фигурных скобок.

```
data = {  
    title: "Sketches of Spain",  
    artist: "Miles Davis"  
};  
template = "Title: {{title}} Artist: {{artist}}";  
result = Mustache.render(template, data);  
console.log(result);
```

```
>Title: Sketches of Spain Artist: Miles Davis
```

Шаблонизатор

Задача шаблонизатора (template engine) подставить все данные из модели в шаблон. В процессе подстановки происходит связывание (binding) имен тегов `{{name}}` с именами свойств json объекта. При изменении шаблона необходимо дополнить модель данными и наоборот при изменении данных нужно расширить шаблон.

Рассмотрим базовые теги шаблонизатора **mustache.js**

Переменные

Основным тегом является переменная. Тег `{{ title }}` генерирует значение имени свойства в объекте данных. Если данных не будет то данный тег ничего не возвращает. Значение `undefined`, `null`, `false`, или `""` не отображаются.

```
data = {  
  title: "Sketches of Spain",  
  artist: ""  
};  
template = "Title: {{title}} Artist: {{artist}}";  
result = Mustache.render(template, data);  
console.log(result);
```

```
>Title: Sketches of Spain Artist:
```

Переменные

```
data = {  
    album: {  
        title: "Blue Train",  
        artist: "John Coltrane"  
    }  
};  
  
template = "Title: {{album.title}}" +  
"Artist: {{album.artist}}";  
result = Mustache.render(template, data);
```

```
>Title: Blue Train Artist: John Coltrane
```


Секции

Секция начинается с- # и заканчивается /. В данном примере {{#person}} — начало секции в то время как {{/person}} — ее окончание. Поведение секции зависит от значения ключевого поля. Если поле "person" имеет значение false или пусто, то HTML внутри секции не будет генерироваться.

```
template = "Shown. {{#person}} <b>Never shown!  
</b>{{/person}}"  
data = {"person": false}  
result = Mustache.render(template, data);  
console.log(result)  
>Shown.
```

Списки

Когда значение свойства список, то текст в блоке будет отображаться для каждого объекта в списке. Контекст в блоке будет устанавливаться для каждой итерации. В итоге мы получим цикл по коллекции объектов.

```
template = "{{#repo}}  
    <b>{{name}}</b>  
    {{/repo}}";  
data = {  "repo": [  
    { "name": "resque" },  
    { "name": "hub" },  
    { "name": "rip" }  
  ]}  
result = Mustache.render(template, data);  
console.log(result);  
> <b>resque</b><b>hub</b><b>rip</b>
```

Рендер с проверкой - ?

Если значение не false и свойство не список то будет единичная генерация блока.

Template:

```
{{#person?}}  
  Hi {{name}}!  
{{/person?}}
```

Data:

```
{  
  "person?": { "name": "Jon" }  
}
```

Output:

Hi Jon!

Инверсия секции

Инверсная секция `{{^repo}}` генерирует значение в том случае если ключ объекта не задан либо значение его `false` либо пустой список `[]`.

```
template:
{{#repo}}
  <b>{{name}}</b>
{{/repo}}
{{^repo}}
  No repos :(
{{/repo}}
data:
{
  "repo": []
}
```

Output:

```
No repos :(
```

Лямбда

Когда значением свойства является callback функция то ей передается контекст родительского контейнера (его внутреннее содержимое). В свою очередь внутри функции происходит вызов рендера для шаблонов в данном контексте.

```
template:
{{#wrapped}}
  {{name}} is awesome.
{{/wrapped}}
data:
{  "name": "Willy",
  "wrapped": function() {
    return function(text, render) {
      return "<b>" + render(text) + "</b>"
    }
  }}
}}
```

Output:

```
<b>Willy is awesome.</b>
```

Комментарии

Комментарии начинаются в тегах с восклицательного знака и игнорируются.

Шаблон:

```
<h1>Today{{! ignore me }}.</h1>
```

Вывод:

```
<h1>Today.</h1>
```

Типичный шаблон

Hello {{name}}

You have just won {{value}} dollars!

{{#in_ca}}

Well, {{taxed_value}} dollars, after taxes.

{{/in_ca}}

Given the following hash:

```
{  
  "name": "Chris",  
  "value": 10000,  
  "taxed_value": 10000 - (10000 * 0.4),  
  "in_ca": true  
}
```

Will produce the following:

Hello Chris

You have just won 10000 dollars!

Well, 6000.0 dollars, after taxes.

Задачи

1. Сформировать список меню с помощью добавления контейнеров `` в `` ``