**Инвариантная самостоятельная работа (ИСР)**

**Развертывание дистанционного ресурса с помощью node.js-платформы Ghost с использованием серверного шаблонизатора Handlebars.**

|  |  |
| --- | --- |
| **Ghost** | |
| [Ghost blogging platform logo.png](https://ru.wikipedia.org/wiki/%D0%A4%D0%B0%D0%B9%D0%BB:Ghost_blogging_platform_logo.png) | |
| **Тип** | Система управления содержимым, блог-платформа |
| **Автор** | Джон О'Нолан, Ханна Вульф |
| **Разработчики** | Ghost Foundation |
| **Написана на** | JavaScript |
| **Операционная система** | Кроссплатформенное ПО |
| **Первый выпуск** | 14 октября 2013 |
| **Последняя версия** | 1.21.3 (21 февраля 2018) |
| **Бета-версия** | 0.3.0 (Сентябрь, 2013) |
| **Состояние** | Активное |
| [**Лицензия**](https://ru.wikipedia.org/wiki/%D0%9B%D0%B8%D1%86%D0%B5%D0%BD%D0%B7%D0%B8%D1%8F_%D0%BD%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5) | [MIT](https://ru.wikipedia.org/wiki/%D0%9B%D0%B8%D1%86%D0%B5%D0%BD%D0%B7%D0%B8%D1%8F_MIT) |
| **Сайт** | [ghost.org](https://ghost.org/) |

Ghost - это бесплатная платформа для ведения блогов с открытым исходным кодом, написанная на JavaScript и распространяемая по лицензии MIT , предназначенная для упрощения процесса онлайн-публикации для отдельных блоггеров, а также онлайн-публикаций.

Концепция платформы Ghost была впервые опубликована в ноябре 2012 года в блоге основателя проекта Джона О'Нолана.

Первая общедоступная версия Ghost, выпущенная в октябре 2013 года, была профинансирована успешной кампанией на Kickstarter, которая достигла своей первоначальной цели финансирования в 25 000 фунтов стерлингов за 11 часов, а затем собрала итоговую сумму в 196 362 фунтов стерлингов в течение 29-дневной кампании.

Идея платформы Ghost была впервые описана в начале ноября 2012 года в блоге основателя проекта Джона О'Нолана  бывшего заместителя руководителя группы по пользовательскому интерфейсу [WordPress](https://en.wikipedia.org/wiki/WordPress" \o "WordPress),  после неудачи в связи со сложностью использования WordPress в качестве [блога,](https://en.wikipedia.org/wiki/Blog) а не [системы управления контентом](https://en.wikipedia.org/wiki/Content_management_system) .

19 сентября 2013 г. - была выпущена первая публичная версия Ghost, названная Kerouac  в качестве раннего выпуска для людей, которые поддержали кампанию Kickstarter.

14 октября 2013 г. - Ghost впервые был доступен для широкой публики в виде расширенного выпуска через GitHub версии 0.3.3 - с исправлениями ошибок и обновлениями безопасности.

Программное обеспечение Ghost для блогов можно загрузить и использовать бесплатно. Кроме того, Фонд предлагает (платную) размещенную платформу для пользователей, которые хотели бы вести живой блог в Интернете, в качестве альтернативы настройке сервера и запуску установки программного пакета вручную. За ежемесячную плату пользователи получают учетную запись с полностью управляемыми блогами Ghost с автоматическим резервным копированием и обновлениями, а также поддержкой по электронной почте.  Поскольку хостинговая платформа принадлежит и управляется Ghost Foundation, весь доход, полученный от сервиса, впоследствии используется для финансирования дальнейшей разработки программного обеспечения с открытым исходным кодом и инфраструктуры проекта.

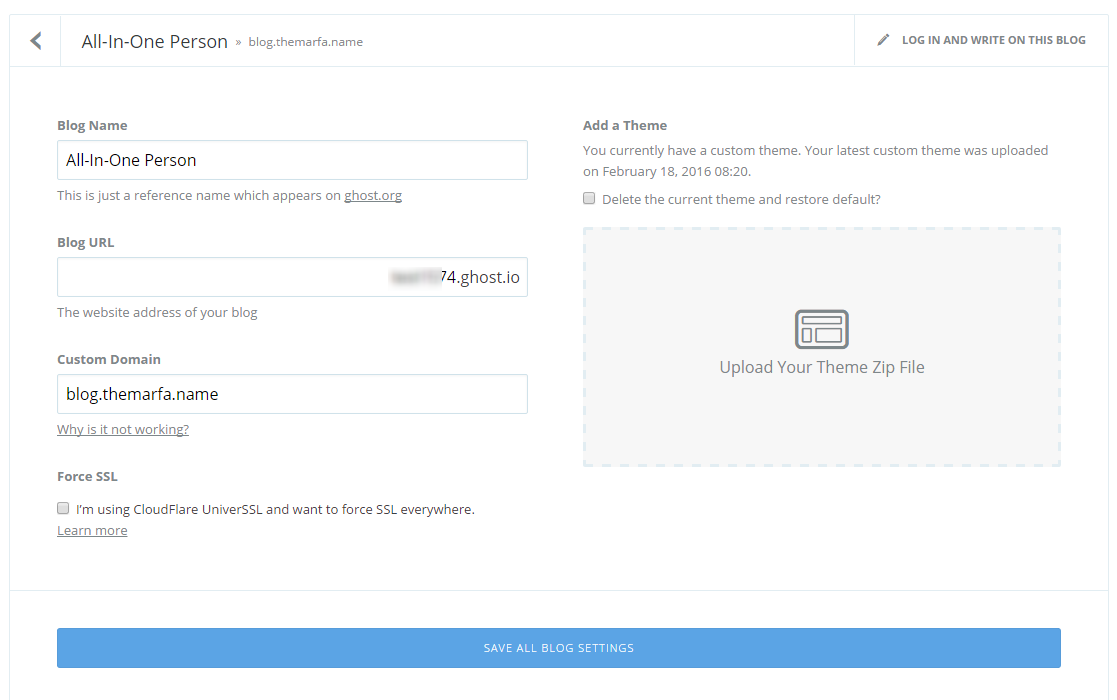
#### Достоинства

1. Open-source движок;
2. Лёгкость настройки;
3. Минималистичный интерфейс;
4. Возможность использовать облачную версию.

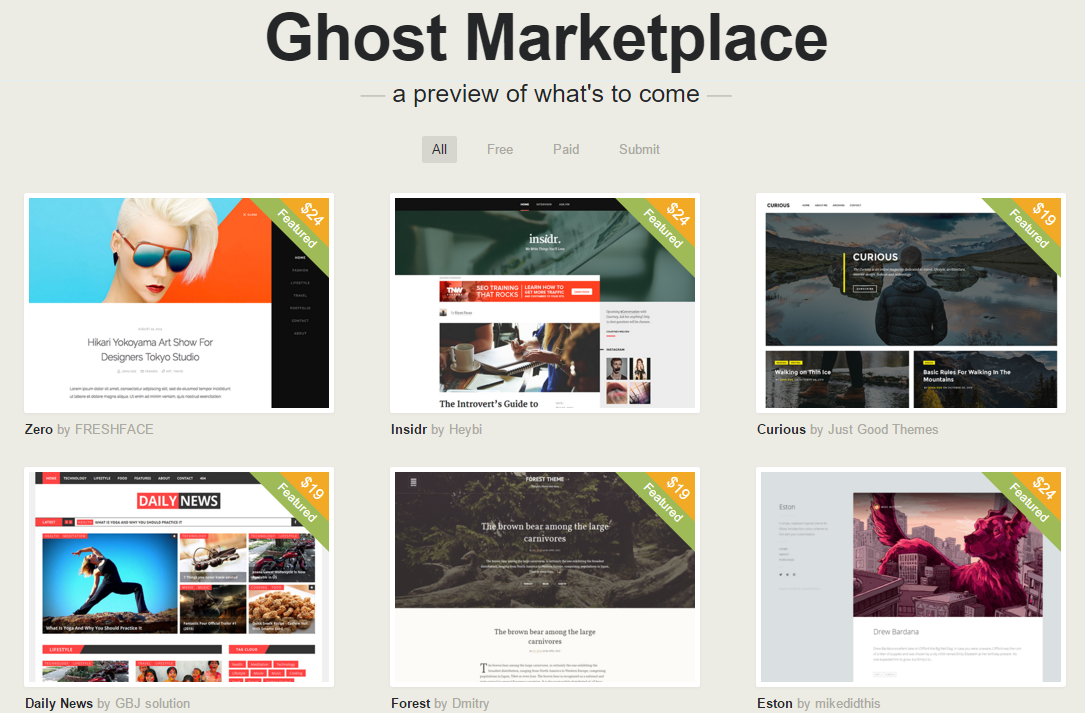
#### Недостатки

1. Редактор темы на GitHub;
2. Подвисания при обновлении кода;
3. Сложности с обновлением при установке на свой сервер.

### Настройка

Новый блог создаётся за секунду. Всё, что нужно указать — адрес блога на сервере Ghost и указать свой домен, если нужно. 

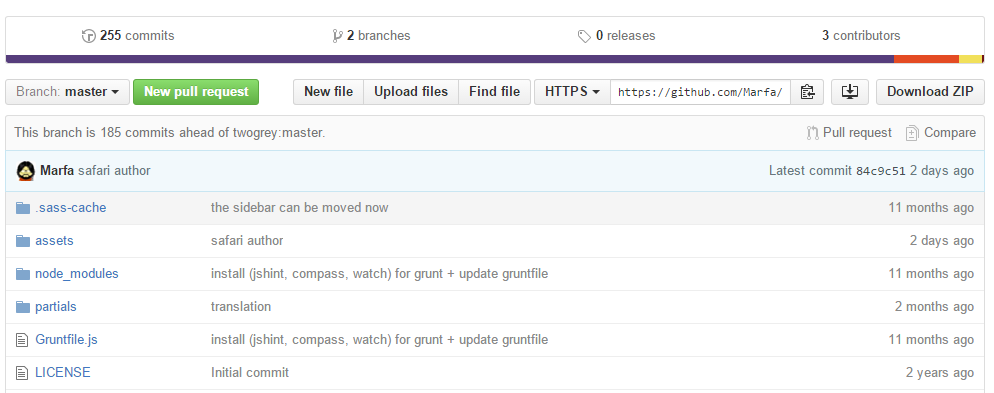
Изначально будущий блог будет основан на стандартной теме, но можно выбрать любую [из магазина тем](http://marketplace.ghost.org/) или купить её на стороннем сайте.



Магазин тем Ghost

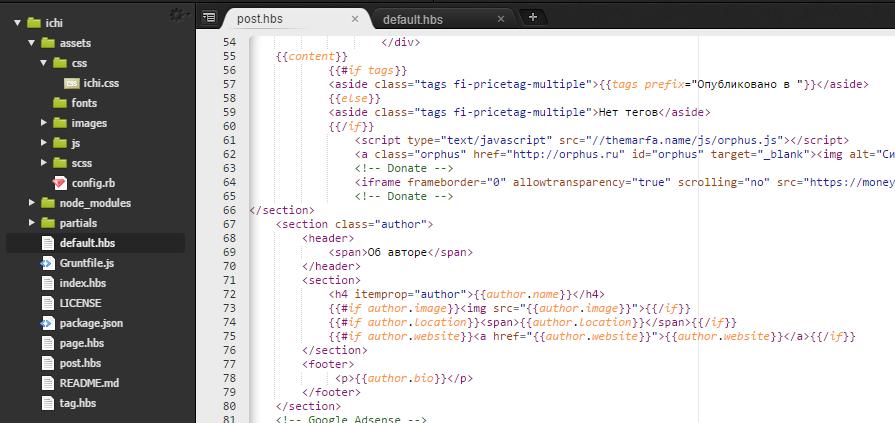
Чтобы обновить тему или загрузить какие-то изменения кода, придётся каждый раз идти в настройки блога и загружать .zip архив с темой. Это очень неудобно и отнимает по полминуты лишнего времени на каждый тест.

Создать или даже отредактировать уже готовую тему сможет далеко не каждый. Платформа по умолчанию настроена на использование [GitHub](https://github.com/" \o "GitHub) в качестве репозитория.

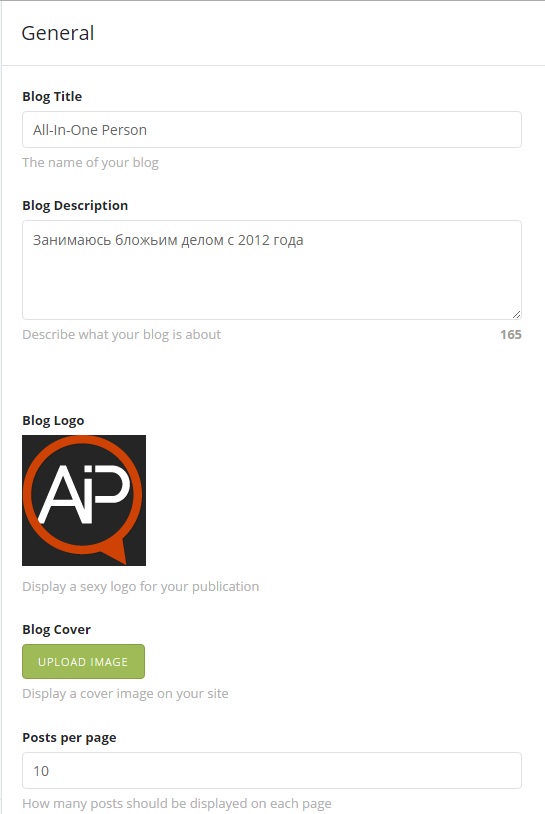


С другой стороны, с кодом и параметрами Ghost для отображения блога разобраться довольно просто, имея даже начальный опыт в веб-разработке. К тому же для разработчиков доступна очень подробная [инструкция](http://support.ghost.org/developers), дружелюбная и быстрая тех.поддержка и не менее дружелюбное сообщество в [Slack](https://ghost.org/slack/" \o "Slack).

Сама тема состоит из нескольких вспомогательных файлов и дополнительных наборов скриптов, которые в конечном счёте рендерят будущий блог:



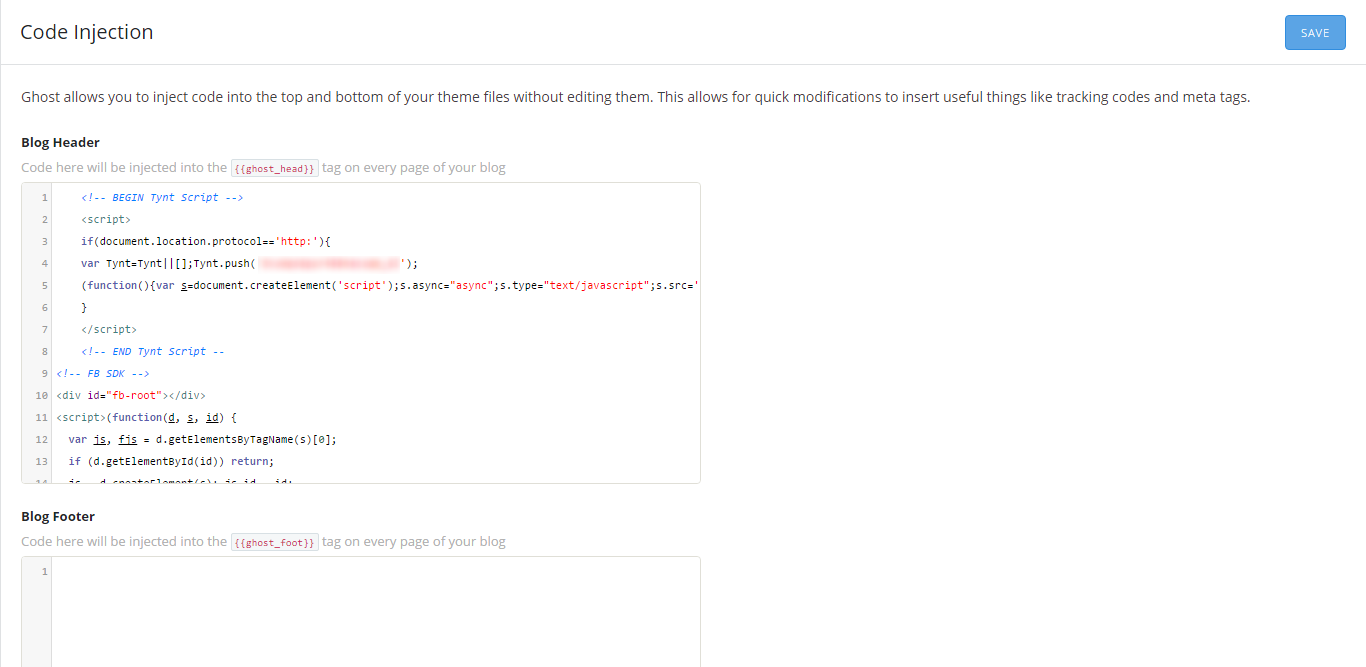
Ну, и конечно, часть базовых настроек доступна прямо из панели управления блогом:



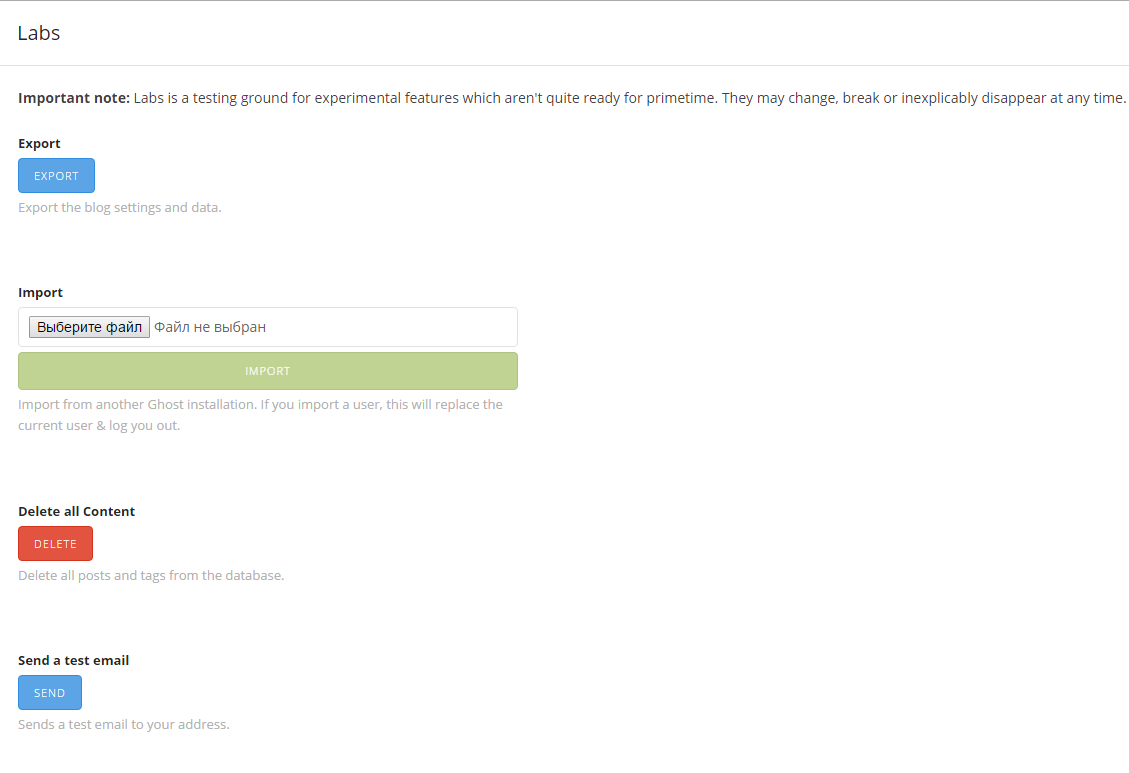


Настройки блога

Чтобы постоянно не обновлять код и не перезагружать тему, в Ghost есть раздел для инъекций. Здесь вы можете сразу же добавить код в шапку или подвал блога.



И, пожалуй, самое главное для любого блогера. Вы можете в любой момент экспортировать или импортировать свои данные. Поэтому с переездом между популярными сервисами не должно возникнуть никаких проблем.



Использование шаблонизатора на стороне клиента просто необходимо при разработке SinglePage-приложений с несколькими представлениями, если ваше приложение является интерактивным и активно отвечает на действия пользователя, наконец, если вы просто хотите отделить разметку от кода, сделать процесс отображения данных более удобным, а код понятным и простым для сопровождения.

Рассмотрим пример формирования HTML из данных без использования шаблонизатора:

JavaScript

bookListHtml = '<ul>';

books.forEach(function(book) {

bookListHtml += '<li class="book">'

+ '<a href="/' + book.code + '">' + book.title

+ ' Author: ' + book.author + '</a></li>';

});

bookListHtml += '</ul>';

console.log(bookListHtml);

Очень часто такой код ведет к дублированию, не говоря уже о том, что поддержка и доработка его становится настоящим испытанием с ростом приложения.

Современное обилие JavaScript-шаблонизаторов позволяют выбрать любой в зависимости от предпочтений, скорости рендеринга, зависимостей, возможностей и т.д.

***Handlebars.js*** — это клиентский шаблонизатор для JavaScript (однако ничего не мешает использовать его на стороне сервера). Он принимает на вход любую строку, состоящую из HTML-тегов и специальных выражений и компилирует их в функцию JavaScript. Эта функция в свою очередь принимает один параметр — объект данных и возвращает строку HTML, где определенные свойства объекта с данными уже вставлены в нужные места шаблона.

Handlebars является одним из наиболее популярных, быстрых и многофункциональных шаблонизаторов для JavaScript. Он практически не позволяет добавлять логику и произвольный JavaScript в шаблоны (кроме циклов и условных выражений, о которых мы поговорим чуть ниже), тем самым он как бы «заставляет» разработчика содержать логику отдельно от разметки.

Handlebars используют такие JavaScript-фреймворки, как Meteor.js, Derby.js, Ember.js, также с ним отлично взаимодействуют и другие фреймворки, например Backbone.js.

**Использование Handlebars.js**

Чтобы начать использовать Handlebars в своем проекте, необходимо подключить файл handlebars.js. Самый простой способ определения шаблона Handlebars — внедрение его непосредственно на страницу внутри тегов <script>.

Markup

<[script](http://docs.webplatform.org/wiki/html/elements/script) [id](http://docs.webplatform.org/wiki/html/attributes/id)="header" [type](http://docs.webplatform.org/wiki/html/attributes/type)="text/x-handlebars-template">

<div>Title: {{title}}</div>

</[script](http://docs.webplatform.org/wiki/html/elements/script)>

Переменные шаблона, а также выражения заключаются в двойные фигурные скобки {{...}}. Данные в шаблон передаются в виде обычного объекта, переменным шаблона соответствуют свойства этого объекта. И наконец, HTML-структура с внедренными данными формируется функцией Handlebars.compile().

Рассмотрим, пример:

Markup

<[script](http://docs.webplatform.org/wiki/html/elements/script) [id](http://docs.webplatform.org/wiki/html/attributes/id)="book" [type](http://docs.webplatform.org/wiki/html/attributes/type)="text/x-handlebars-template">

<h2>{{bookTitle}}</h2>

<div>Author: {{bookAuthor}}</div>

</[script](http://docs.webplatform.org/wiki/html/elements/script" \t "_blank)>

JavaScript

// Инициализируем объект с данными

var book = { bookTitle: '2001: A Space Odyssey', bookAuthor: 'Arthur Clarke' };

// Получаем шаблон

 var templateScript = $('#book').html();

// Функция Handlebars.compile принимает шаблон и возвращает новую функцию

 var template = Handlebars.compile(templateScript);

// Формируем HTML и вставляем в документ

$(document.body).append(template(book));

**Синтаксис Handlebars.js**

Выражения Handlebars, как мы уже говорили, заключаются в двойные фигурные скобки. Выражением может быть как переменная, так и [хелпер](https://getinstance.info/articles/tools/custom-handlebars-helpers/) — предварительно определенная функция, результат которой будет вставлен в шаблон.

Также Handlebars поддерживает несколько [встроенных хелперов](https://getinstance.info/articles/tools/standart-handlebars-helpers/), или блоков, каждый из которых открывается символами «{{#», а закрывается символами «/}}». Про встроенные хелперы мы еще поговорим подробнее, примером такого блока может служить блок if:

JavaScript

{{#if someValueI}}  Content  {{/if}}

Комментарии Handlebars заключаются в символы {{! }} (ну и конечно же, вы можете использовать обычные HTML-комментарии).

JavaScript

{{! Comment content }}

Если свойство объекта данных в свою очередь тоже является объектом, можно вставлять в шаблон значения его свойств через точку, например:

JavaScript

// Объект с данными

var data = { name: { firstName: 'John', lastName: 'Smith' } }

// Шаблон

<div>{{name.firstName}}</div>

Если свойство объекта данных является массивом, можно реализовать вывод элементов массива с помощью цикла each:

JavaScript

var data = {

groupName: 'Customers',

users: [

{ name: { firstName: 'John', lastName: 'Smith' }},

{ name: { firstName: 'Thomas', lastName: 'Anderson' }}

]

};

Markup

<[script](http://docs.webplatform.org/wiki/html/elements/script) [id](http://docs.webplatform.org/wiki/html/attributes/id)="users-template" [type](http://docs.webplatform.org/wiki/html/attributes/type)="x-handlebars-template">

{{#each users}}

<li>

{{name.firstName}} {{name.lastName}} is in the {{../groupName}} group.

</li>

{{/each}}

</[script](http://docs.webplatform.org/wiki/html/elements/script" \t "_blank)>

На пример выше стоит обратить особое внимание. Здесь при заходе в цикл меняется объект данных (в терминах Handlebars он называется контекстом). Вернуться в родительский контекст позволяет выражение «../».

Handlebars экранирует любые HTML-строки, переданные в шаблон. Чтобы вывести исходный HTML, необходимо использовать тройные фигурные скобки {{{ ... }}}.

Рассмотрен только базовый синтаксис Handlebars.js, необходимый для того, чтобы начать его использовать. На самом деле, этот шаблонизатор имеет массу возможностей.