УДК 004.42(07)
Н.А. Христинець, Я.П. Цяпич
Луцький національний технічний університет

HAML/SASS ЯК РОЗШИРЕННЯ ДЛЯ HTML/CSS ПРИ ВЕРСТЦІ ВЕБ-САЙТІВ

Н.А. Христинець, Я.П. Цяпич. Haml/sass як розширення для html/css при верстці веб-сайтів. Досліджено актуальність заміни стандартної реалізації гіпертекстової розмітки Html та стилей CSS на спрощену розмітку Haml та скриптову метамову Sass, яка інтерпретується в каскадні таблиці стилів CSS, при написанні клієнської (front end) частини веб-сайту.
Ключові слова: верстка, Haml, Sass, препроцесорна мова, транслювання кодів, розмітка сайта.

Н.А.Христинец, Я.П.Цяпич. Haml/sass как расширение для html/css в верстке веб-сайтов. Исследованы актуальность замены стандартной реализации гипертекстовой разметки Html и стилей CSS на упрощенную разметку Haml и скриптовый метаязык Sass, который интерпретируется в каскадные таблицы стилей CSS, при написании клиентскои (front end) части сайта.
Ключевые слова: вёрстка, Haml, Sass, препроцессорный язык, трансляции кодов, разметка сайта.

N. Hrystynets, J. Tsyapych. Haml / sass as an extension to html / css layout with websites. In this research work was investigated the relevance replace the standard implementation of Hypertext Markup Html and sheets CSS at sproschynu Haml markup and scripting metalanguage Sass, which is interpreted in Cascading Style Sheets CSS, in writing clients' (front end) of the website.
Keywords: layout, Haml, Sass, preprocessing language translation codes, marking the site.

Постановка проблеми у загальному вигляді і її зв'язок з важливими науковими та прктичними завданнями. Haml (xHTML Abstraction Markup Language) – мова розмітки для спрощеної генерації xHTML. В свою чергу, еквівалент Haml для css – це Sass (Syntactically Awesome StyleSheets). Haml представляє з себе мову розмітки з спрощеним синтаксисом і написаний на ньому код може бути скомпільований в HTML. Як і інші препроцесори, він пропонує не тільки видозмінений синтаксис, але й нові конструкції мови: умовні оператори, цикли, вставки коду з інших файлів тощо. В свою чергу Sass – це мова, схожа до Haml, але призначена для спрощення створення CSS-коду. Код Sass спеціальної ruby-програмою транслюється в звичайний CSS код. Синтаксис цієї мови дуже гнучкий, він враховує безліч дрібниць, які так бажані у CSS. Більш того, в ньому є навіть логіка (if, each), математика (можна складати як числа, рядки, так і кольори).

Актуальність проблеми формування клієнтської частини веб-сайту грунтується на тому, що використання Haml и Sass є набагато ефективнішими і необхідно, враховуючи, що частина роботи, яка повинна була б реалізовуватись на серверній частині, так званій backend розробці, може виконувати безпосередньо браузерним інтерпретатором, при чому код реалізації є більш лаконічнім, зрозумілим і моноструктурним.

Проблемою є те, що при стандартній верстці макету сайту, при використанні стандартної розмітки, ми отримуємо багато коду і при цьому ж не можемо зробити нашу html-сторінку більш динамічною та інтерктивною в функціональному плані. Багато нововведень, які досі зв'язували руки верстальщикам при написанні клієнтської частини сайту, були розроблені і впроваджені в препроцесорних мовах, одними із яких є Haml і Sass. Ще одним аспектом проблеми є економія сил і часу, що є особливо важливо у роботі веб-студій, коли час виконання front end-частини проекту повинна виконуватись якомога швидше, щоб передати розробку веб-сайта в руки тих, хто програмуватиме функціонал (серверну частину).

Тому, виникнення таких технологій достатньо успішно вирішує проблеми, пов'язані з часом на виконання верстки, та ще й додають нові можливості, які розширюють функціональність мов, що використовуються під час верстки веб-сайтів.

Ціль статті. Метою даної роботи є демонстрування на прикладі простоти, лаконічності, додаткових можливостей та нового рівня абстрагування в Html-розмітці та в каскадних стилях CSS при верстці веб-сайту. Головним завданням є дослідити доцільність та необхідність реалізації в Html та CSS арифметики, циклів та змінних при розробці користувацької частини проекту сайту.

Виклад основного матеріалу дослідження. У даній роботі зображено порівняльний код, який демонструє спрощений синтаксис та появу нових можливостей в Haml/Sass [1,2]. Для початку наведем простий код Haml в порівнянні з Html:

Haml код:	Після компіляції отримаємо наступний html код:
#wrapper %section#content.main-content %header.header %h1 Заголовок h1 .post-date 28.08.2014 .post-entry %p Перший абзац тексту. %p Другий абзац тексту.	 <section class="main-content" id="content"> <header class="header"> Заголовок h1 28.08.2014 </header> Перший абзац тексту. Другий абзац тексту. </section>

Перше, що заслуговує уваги - відсутність закриваючих тегів. Друге - замість закриваючих тегів використовуються відступи. Саме за допомогою табуляції визначається вкладеність. Імена тегів починаються з символу%, далі йде ім'я тега і його параметри class / id, якщо вони необхідні. Якщо ім'я тега опущено, як у випадку з блоком #wrapper, то за замовчуванням буде використовуватися тег div. Як бачимо, багато чого направлено на скорочення кількості коду і це має суттєву перевагу, так як нові можливості додають функціональності сайту.

Заслуговує увагу можливість використання в розмітці змінних, циклів, умовних операторів, масивів тощо. Оголошення змінної і приведений нижче код демонструє наступний приклад:

- some_var = 'Значення змінної' %p = some_var	На виході отримуємо: Значення змінної

З використанням цикла і оператора if, конструкція буде виглядати наступним чином:

.nav.nav-pagination %ol.pages - (1..6).each do |i|	Компілюємо і отримуємо: <ol class="pages">

```
%li.page
 - if i == 3
   %span.current #{i}
 - else
   %a{:href => '#'} #{i}
```

```
<li class="page">
  <a href="#">1</a>
</li>
<li class="page">
  <a href="#">2</a>
</li>
<li class="page">
  <span class="current">3</span>
</li>
<li class="page">
  <a href="#">4</a>
</li>
<li class="page">
  <a href="#">5</a>
</li>
<li class="page">
  <a href="#">6</a>
</li>
</ol>
</div>
```

Якщо раптом буде потрібно збільшити або зменшити кількість сторінок, змінити імена класів або додати додаткову розмітку, достатньо буде декількох невеликий правок. В цьому полягає мобільність і гнучкість вибраного застосування [3]. Одними із нововведень в Sass, які можна відмітити окремо, є вкладеність, наслідування, інтерполяція і підстановка.

```
h1{
  font-size: 12px;
  color: #ccc;
}
h1 a{
  text-decoration: none;
  color: #ddd;
}
```

Тобто, пишемо, що заголовку першого рівня необхідно привласнити властивості - кегль і колір шрифту. Посиланню, вкладеному у цей заголовок, присвоюємо властивості - прибираємо підкреслення і задаємо свій колір.

Тепер запишемо той самий код в Sass:

```
h1{
    font-size: 12px;
    color: #ccc;
    a{
      text-decoration: none;
      color: #ddd;
    }
  }
```

Особливістю такого варіанту є те, що на Sass ми пишемо код так, як бачимо. Дивимося HTML-код - посилання вкладено у тег h1. Тому, в Sass-коді так і записуємо – всередині селектора h1 поміщаємо селектор a, кажучи тим самим Sass, що тег a вкладений в тег h1. Після конвертації в CSS-код у нас вийде точно такий код, як у першому прикладі (де представлений CSS-код).

Принцип наслідування реалізується наступним чином. За допомогою наслідування можна значно скоротити код і прискорити його написання. Принцип також простий - якщо для одного елемента заданий набір властивостей, то можна для іншого елемента вказати наслідуванню ці ж самі властивості. Наслідування в Sass виконується за допомогою директиви @extend. В наведеному нижче коді, ми бачимо, як заголовок h2 унаслідує властивості заголовка у заголовка h1:

```
h1{
    font: {
      size: 30px;
      style: italic;
      weight: bold;
      family: Arial, sans-serif;
    }
   color: #ccc;
   padding-left: 10px;
  }
  h2{
   @extend h1;
   color: #aaa;
  }
```

Під інтерполяцією тут мається на увазі підстановка значення змінної в назву CSS-властивості. Завдяки цьому, можна легко змінювати саму властивість у функції. Наприклад, таким чином:

```
$radius: 8px;
$vert: top;
$hor: left;

@mixin corners {

        Border-#{$vert}-#{$hor}-radius: $radius;

}
```

Підстановка (Mixin) в Sass повністю відповідає своїй назві за принципом застосування, але, в той же час, нагадує функцію. Тобто, Sass ще більше розширює можливості CSS і частково перетворює його з описової мови в мову програмування. Після оголошення mixin його підключають в потрібному місці коду за допомогою директиви @include. Зручність застосування підстановки полягає в тому, що не потрібно щоразу писати один і той самий код в різних місцях [4].

```
@mixin border-radius {
      -webkit-border-radius: $radius;
 -moz-border-radius: $radius;
border-radius: $radius;
}
.box {
height: 200px;
          width: 400px;
background-color: #cc000;
@include border-radius;
}
```

Висновки. Розглянуті вище технології дозволяють значно спростити написання важливих елементів веб-сайта. Вони розширюють можливості використання в «фронт енді» функціональності верстки за допомогою циклів, змінних, функцій тощо. При цьому стисліть, лаконічність і зрозумілість коду виходить на новий рівень. Актуальність даних технологій постійно зростає, адже вони не тільки додають нових можливостей і забезпечують прозорість коду, але й суттєво економлять час при написанні клієнтської частини веб-сайту.

Список використаних джерел.

1. Брайан Хоган, К. Уоррен, М. Уэбер, К. Джонсон, А. Годин // Книга веб-программиста: секреты профессиональной разработки веб-сайтов. - Питер. - С. 197-203. - 288 с.
2. Зольников Д.С. PHP 5. Как самостоятельно создать сайт любой сложности. - М.: НТ Пресс, 2005. - 109 с.
3. Мак Т., Вест Р. Dreamweaver MX 2004. Шаг за шагом. Самоучитель. - М.: ЭКОМ, 2006. - 312 с.
4. Хестер Н. Создание Web-страниц в Dreamweaver. - М.: НТ Пресс, 2005. - 104 с.
5. Кристофер Шмитт. CSS. Рецепты программирования = CSS. - СПб.: БХВ-Петербург, 2007. - 592 с.
6. Энди Бадд, Камерон Молл, Саймон Коллизон. Мастерская CSS: профессиональное применение Web-стандартов. - М.: Вильямс, 2007. - 272 с.