

Ускорение разработки клиентских приложений

HTML -> HAML

CSS -> LESS, SASS

JavaScript -> CoffeeScript

HAML

Haml ([HTML](#) abstraction markup language)

Haml - легковесный язык разметки, который используется для описания HTML-документа.

Основные принципы HAML:

- * Разметка должна быть красивой
- * Разметка не должны иметь повторений(DRY)
- * Разметка должна иметь отступы
- * Структура HTML должна быть понятной

HAML

Haml преобразуется в HTML с использованием парсеров. Существуют парсеры на Ruby, Python, PHP, JavaScript, ...

Парсеры для PHP:

- * pHAML - <http://phaml.sourceforge.net/>
- * phpHAML - <http://phphaml.sourceforge.net/>
- * Multi target HAML - <https://github.com/arnaud-lb/MtHaml>

HAML

Основы синтаксиса Haml:

- * имена тэгов начинаются с символа %, далее следует имя тэга и его параметры class(начинается с .)/id (начинается с #), если они необходимы. Если имя тэга опущено то будет использоваться div.
- * вложенность определяется с помощью табуляции - вместо закрывающих тэгов используются отступы.

HAML

```
#wrapper
```

```
%section#content.main-content
```

```
%header.header
```

```
%h1 Заголовок h1
```

```
.post-date 28.08.2015
```

```
.post-entry
```

```
%p Содержимое параграфа
```

```
%p Содержимое параграфа 2
```

```
<div id="wrapper">
```

```
  <section id="content" class="main-  
  content">
```

```
    <header class="header">
```

```
      <h1>Заголовок h1</h1>
```

```
      <div class="post-date">28.08.2015</div>
```

```
    </header>
```

```
    <div class="post-entry">
```

```
      <p>Содержимое параграфа</p>
```

```
      <p>Содержимое параграфа 2</p>
```

```
    </div>
```

```
  </section>
```

```
</div>
```

HAML

Объявление переменной и вывод на печать:

```
- some_var = 'Значение переменной'  
%p  
  = some_var
```

На выходе получаем:

```
<p>  
  Значение переменной  
</p>
```

HAML

Условный оператор If

- myVar = 9
- if myVar == 10
 - %p Значение переменной равно 10
- else
 - %span Значение переменной НЕ равно 10

HAML

Циклы в Haml

```
.nav.nav-pagination
```

```
  %ol.pages
```

```
    - for ($i=1;$i<3; $i++)
```

```
      %li.page
```

```
        - if $i == 3
```

```
          %span.current $i
```

```
        - else
```

```
          %a{:href => '#'} $i
```


!!! 1.1

%html

%head

**%meta{ :http-equiv => 'Content-Type', :content =>
'application/xhtml+xml;charset=utf-8' }**

- if (\$title)

%title= \$title

- else

%title= \$pagename

%body

#header

%h1 Example page

- if (\$slogan)

%span= \$slogan

#content

%table.config.list

%tr

%th ID

%th Name

%th Value

- foreach (\$config as \$c)

%tr[\$c]

%td= \$c->ID

%td= \$c->name

%td= \$c->value

#footer

%span.author John Random

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"

"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<html>

<head>

<meta http-equiv="Content-Type"

content="application/xhtml+xml;charset=utf-8" />

<?php if (\$title) { ?>

<title><?php echo \$title; ?></title>

<?php } else { ?>

<title><?php echo \$pagename; ?></title>

<?php } ?>

</head>

<body>

<div id="header">

<h1><?php echo \$pagename; ?></h1>

<?php if (\$slogan) { ?>

<?php echo \$slogan; ?>

<?php } ?>

</div>

<div id="content">

<table class="config list">

<tr><th>ID</th><th>Name</th><th>Value</th></tr>

<?php foreach (\$config as \$c) { ?>

<tr class="<?php echo (\$class = forClassName(\$c)); ?>"

id="<?php echo "\$class_{ \$c->ID}"; ?>">

<td><?php echo \$c->ID; ?></td>

<td><?php echo \$c->name; ?></td>

<td><?php echo \$c->value; ?></td>

</tr>

<?php } ?>

</table>

</div>

<div id="footer">

John Random

</div>

</body>

</html>

SASS

Sass (Syntactically Awesome Stylesheets) — это метаязык на основе CSS, предназначенный для увеличения уровня абстракции CSS кода и упрощения файлов каскадных таблиц стилей.

Язык Sass имеет два синтаксиса:

- * **sass** — отличается отсутствием фигурных скобок, в нём вложенные элементы реализованы с помощью отступов (как в HAML);
- * **scss (Sassy CSS)** — использует фигурные скобки, как и сам CSS.

SASS

Вложенные правила

```
#main p {  
  color: #00ff00;  
  width: 97%;
```

```
  .redbox {  
    background-color: #ff0000;  
    color: #000000;  
  }  
}
```

```
#main p {  
  color: #00ff00;  
  width: 97%;  
}
```

```
#main p .redbox {  
  background-color: #ff0000;  
  color: #000000;  
}
```

SASS

Ссылка на родительский элемент селектора

```
a {  
  font-weight: bold;  
  text-decoration: none;  
  &:hover { text-decoration: underline; }  
  body.firefox & { font-weight: normal; }  
}
```

```
a {  
  font-weight: bold;  
  text-decoration: none;  
}  
a:hover {  
  text-decoration: underline;  
}  
body.firefox a {  
  font-weight: normal;  
}
```

SASS

Вложенные свойства

```
.funky {  
  font: {  
    family: fantasy;  
    size: 30em;  
    weight: bold;  
  }  
}
```

```
.funky {  
  font-family: fantasy;  
  font-size: 30em;  
  font-weight: bold;  
}
```

SASS

SassScript допускает использование *переменных*. Переменные начинаются со знака доллара \$ и задаются как свойства CSS:

```
$width: 5em;
```

Можно обратиться из свойств:

```
#main {  
  width: $width;  
}
```

Переменные доступны только в пределах того уровня вложенности селекторов, на котором они определены. Если они определяются вне каких-либо вложенных селекторов, они доступны глобально. Можно определить переменную со специальной меткой !global, чтобы переменная также была доступна глобально. Например:

```
#main {  
  $width: 5em !global;  
  width: $width;  
}  
#sidebar {  
  width: $width;  
}
```

SASS

SassScript поддерживает семь основных **типов данных**:

1. числа (1.2, 13, 10px)
2. текстовые строки, с кавычками и без них ("foo", 'bar', baz)
3. цвета (blue, #04a3f9, rgba(255, 0, 0, 0.5))
4. булевы значения (true, false)
5. null
6. списки значений, с разделительными пробелами или запятыми (1.5em 1em 0 2em; Times New Roman, Arial, sans-serif)
7. массивы(мапы) (key1: value1, key2: value2)

SassScript также поддерживает все другие виды данных CSS, такие как диапазоны Unicode и декларации !important, однако для них не имеется специальных способов обращения, они используются точно также, как строки без кавычек.

SASS

Пример использования различных типов

```
$status-colors: (  
  primary: #000,  
  success: #27BA6C,  
  info: #03a9f4,  
  warning: #FF8833,  
  danger: #ff1a1a  
);  
  
.message {  
  @each $status, $color in $status-  
colors {  
    &--#{$status} {  
      background: $color;  
    }  
  }  
}  
  
.message--primary {  
  background: #000;  
}  
  
.message--success {  
  background: #27ba6c;  
}  
  
.message--info {  
  background: #03a9f4;  
}  
  
.message--warning {  
  background: #f83;  
}  
  
.message--danger {  
  background: #ff1a1a;  
}
```


SASS

* Операции с числами

SassScript поддерживает стандартные арифметические операции над числами (сложение +, вычитание -, умножение *, деление / и остаток от деления по модулю %). Математические функции Sass сохраняют значение промежуточных результатов в течение выполнения арифметических операций.

```
p { color: #010203 + #040506; } -> p { color: #050709; }
```

```
p { margin: 3px + 4px auto; } -> p { margin: 7px auto; }
```

SASS

* Директивы и правила

Sass расширяет CSS правило **@import**, позволяя импортировать scss и sass файлы. Все импортированные scss и sass файлы могут быть объединены в одном результирующем css файле. Кроме того, любые переменные или миксины, объявленные в импортируемых файлах, могут использоваться в главном файле.

Директива **@extend** сообщает Sass, что один селектор должен наследовать стили другого. Например:

```
.error {border: 1px #f00; background-color: #fdd;}
```

```
.seriousError { @extend .error; border-width: 3px; }
```

компилируется в

```
.error, .seriousError { border: 1px #f00; background-color: #fdd; }
```

```
.seriousError { border-width: 3px; }
```

SASS

* Миксины

Миксины (часто используется название **примеси**) позволяют определить стили, которые могут быть использованы повторно в любом месте документа без необходимости прибегать к несемантическим классам вроде **.float-left**. Миксины также могут содержать целые CSS правила или что-либо другое, разрешённое в Sass документе. Они даже могут принимать аргументы, что позволяет создавать большое разнообразие стилей при помощи небольшого количества миксинов.

SASS

* Миксины

Миксины объявляются директивой `@mixin`. После неё должно стоять имя миксина и, опционально, его параметры, и блок, содержащий тело миксина. Например, можно определить миксин `large-text` следующим образом:

```
@mixin large-text {  
  font: { family: Arial; size: 20px; weight: bold; }  
  color: #ff0000;  
}  
  
.page-title {  
  @include large-text;  
  padding: 4px;  
  margin-top: 10px;  
}
```

COFFEE SCRIPT

CoffeeScript — язык программирования, транслируемый в JavaScript. CoffeeScript добавляет JavaScript синтаксический сахар в духе Ruby, Python, Haskell и Erlang для того, чтобы улучшить читаемость кода и уменьшить его размер.

CoffeeScript позволяет писать более компактный код по сравнению с JavaScript.