УДК 004.021

Жмуркевич Андрій Євгенович
Львівський національний університет імені Івана Франка
Жмуркевич Андрей Евгеньевич
Львовский национальный университет имени Ивана Франко
Zhmurkevych Andrey
Ivan Franko National University of Lviv

Мигаль Вікторія Віталіївна
студент
Львівський національний університет імені Івана Франка
Мигаль Виктория Витальевна
студент
Львовский национальный университет имени Ивана Франко
Myhal Viktoriia
student
Ivan Franko National University of Lviv

ВИКОРИСТАННЯ ПРЕПРОЦЕСОРА SASS ДЛЯ ПОБУДОВИ ВЕБ-СТОРІНОК З ДОПОМОГОЮ GULP.JS

ИСПОЛЬЗОВАНИЕ ПРЕПРОЦЕССОРА SASS ДЛЯ ПОСТРОЕНИЯ ВЕБ-СТРАНИЦ С ПОМОЩЬЮ GULP.JS

USING SASS PREPROCESSOR FOR BUILDING WEB PAGES USING GULP.JS

Анотація. *У статті розглядається розробка адаптивних, кросбраузерних веб-сторінок для пристроїв з різним розширенням монітора і з використанням різних веб браузерів для відображення. Для оптимальної організації коду, зменшення кількості помилок при його написанні та для покращення архітектури проекту використовуються препроцесор SASS і система для збірки проектів Gulp.js. У публікації описано причини вибору згаданих технологій та спосіб їх використання у проектах.*

Ключові слова: *веб розробка, верстка, використання препроцесора, система для збірки проектів, оптимізація коду.*

Аннотация. *В статье рассмотрено разработку адаптивных, кросс-браузерную веб-страниц для устройств с различным расширением монитора и с использованием различных веб браузеров для отображения. Для оптимальной организации кода, уменьшение количества ошибок при его написании и для улучшения архитектуры проекта используются препроцессор SASS и система для сборки проектов Gulp.js. В публикации описано причины выбора данных технологий и способ их использования в проектах.*

Ключевые слова: *веб разработка, верстка, использование препроцессора, система для составления проектов, оптимизация кода.*

Abstract. *This article discusses the development of adaptive, cross-browser web pages for devices with different screen resolution and using different browsers to display. For the code organization, to do less error and to enhance architecture project is using technologies such as preprocessor SASS and the streaming build system Gulp.js. The publication describes the reasons for choosing these technologies and using them in project to write high quality, optimized, how don't duplicate code and spread errors when the project will be expand.*

Key words: *web development, mark-up, using preprocessor, streaming build system, code optimization.*

Значне розповсюдження Інтернет послуг різного спрямування, виробництво та продаж пристроїв з різноманітними характеристиками та можливістю роботи в мережі зумовлюють специфічні вимоги до наповнення веб сайтів. Відвідувачі інтернет сторінок очікують коректного та якісного відтворення усього контенту на екрані незалежно від типу пристрою. Складні графічні елементи мають поєднуватися із зручним та зрозумілим функціональним інтерфейсом. Очевидно, що у цьому випадку розробник повинен виконати додаткову роботу щодо адаптації веб сторінки для роботи на різнотипних пристроях, а також врахувати платформу на якій відкривається ця сторінка. Йдеться про те, що часто браузери збудовані з використанням різних систем, а веб аплікації та мобільні додатки мають свої специфічні вимоги до написання коду.

Висвітлимо деякі аспекти верстки веб сторінок. Зазвичай для оформлення нескладного сайту, який виконує лише функцію відображення інформації достатньо використати звичайні CSS стилі. Розширення функціональності, гнучкість в управлінні та крос-браузерність зумовлюють швидке зростання кількості громіздких конструкцій, дублювання коду, що у свою чергу збільшує ризик наявності помилок. Якщо один файл налічує до прикладу кілька тисяч рядків коду, пошук помилки звичайно ускладнюється, особливо у випадку використання простих CSS. Трапляються ситуації, коли для отримання прийнятного результату доводиться здійснювати виправлення в декількох фрагментах коду, які дублюються. Для уникнення цього розробники оптимізують свій код з використанням змінних, циклів, функцій та інших мовних конструкцій. Завдяки спеціальним технологіям така практика розробки стає доступною і у процесі верстки сайтів. Вона стає не просто набором правил для написання стилів, а функціоналом, який потрібно логічно продумати і використати.

Існує значна кількість технологій з відкритим доступом для покращення верстки: фрейморки, бібліотеки, плагіни, препроцесори. Звичайно виокремити якусь конкретну з них як найкращу неможливо. Переваги від застосування тієї чи іншої технології залежать від специфіки проекту, від завдань поставлених розробнику та складності їх реалізації, а також від уподобань самого розробника. На великих проектах сьогодні не часто використовують «чистий» CSS, адже розробка стає доволі тривалою, з'являється необхідність урахування та контролю різних нюансів відображення веб елементів. Для уникнення вказаних проблем варто використати технологію для верстки — препроцесор SASS.

Отже, препроцесор — це надбудова, яка розширяє CSS новими можливостями і синтаксичними конструкціями. Розробник будує Sass-файл, який певним чином обробляється і зберігається у вигляді простого CSS-файлу. Далі згенерований файл можна використовувати на будь-якому сайті. Фактично препроцесор SASS — це прошарок між таблицями стилів, які пишуться і CSS стилями які віддаються браузеру, тобто браузер отримує чистий перевірений код. Основне завдання препроцесора полягає у створені зручних конструкцій для розробника, які спрощують та прискорюють розробку. Отже з'являється можливість писати читабельний, структурований і логічний збудований код, націлений на продуктивне використання.

Наведемо деякі можливості SASS. Уявимо ситуацію, що на сторінці є декілька блоків, їм потрібно задати різний колір фону, а при наведені вказівника миші змінити фон добавивши прозорість.

1. Реалізація за допогою SASS.

```
$main-colors: (0: #4f5154, 1: #ffffff, 2: #4879c6);
$hover-colors: (0: #405154, 1: #f87676, 2: #f87676);
@function getFromObj($key, $obj) {
  @if map-has-key($obj, $key) {
    @return map-get($obj, $key);
  }
  @warn «Unknown ‘#{$obj}’ in $main-colors.»;
  @return null;
}
@mixin modul-color($modul:0){
background: getFromObj ($modul, $main-colors);
&: hover{
    background: getFromObj ($modul, $hover-colors);
  }
}
@each $status, $color in getFromObj(main, $module-
colors) {
  .modul-#{$status} {
    @include modul-color($status);
  }
}
```

2. Реалізація за допомогою CSS:

```
.modul-0 {
    background: #4f5154;
}
.modul-1 {
    background: #ffffff;
}
.modul-2 {
    background: #4879c6;
}
.modul-0: hover{
    background: #405154;
}
.modul-1: hover{
    background: # f87676;
```

```
}
.modul-2: hover{
   background: # f87676;
}
```

На перший погляд написаний за допомогою SASS код більший і може видатися неефективним для виконання вказаного вище простого завдання. Ситуація кардинально змінюється, якщо виникне потреба добавити ще якусь кількість блоків. У цьому випадку SASS код потрібно буде модифікувати лише добавивши в об'єкт кольори, а в CSS дублювати усі необхідні блоки.

В SASS застосовується функція getFromObj(), яка повертає значення об'єкта по ключу. Вказану функцію можна використовувати в різних випадках, до прикладу — зберігши розміри шрифтів, час виконання анімації чи інші дані у вигляді об'єкта і за допомогою однієї функції завжди мати доступ до потрібних властивостей.

SASS дозволяє використовувати багато різних конструкцій, має потужний функціонал. До того ж маємо багато CSS фрейморків і навіть шрифтів написаних за допомогою SASS.

Розгляд синтаксису будь-якої сучасної мови програмування дозволяє відзначити використання змінних, констант, методів, функцій, класів та іншого. Застосування вказаних інструментів дозволяє розробнику дотримуватися принципу «Don't repeat yourself» — «не повторюю себе», тобто варто один раз описати певні шаблони, щоб мати можливість багаторазово їх використовувати. Власне SASS дозволяє типові синтаксичні конструкції мов програмування застосувати до CSS. Завдання полягає лише у правильній компіляції вхідного SASS-коду до вигляду стандартного CSS, який однозначно сприймається браузером.

Очевидно що на сьогодні відомо багато різних способів компіляції, але зупинимося на розгляді інструменту для збірки проектів Gulp.js, який виграє на фоні інших багатофункціональністю та зручністю. Однією з функцій Gulp.js є компіляція коду SASS в код CSS.

При розробці проектів, значну увагу варто приділяти їхній структурі. Для того, щоб правильно сформувати дерево проекту, зазвичай створюють окремі папки для розташування файлів з кодом SASS і кодом CSS. SASS файлів в одній папці може бути багато, але для зручності в подальшому використанні, рекомендовано об'єднати всі SASS файли в один. Отриманий файл компілюється в CSS за допомогою Gulp.js.

Зазвичай невеликий проект містить один CSS файл. Коли функціонал ускладнюється і розширюється, рекомендовано використовувати різні CSS файли, щоб уникнути проблеми завантаження стилів, які не використовуються на активній веб сторінці.

Беручи до уваги вищеописану структуру проекту, можна створити такий файл налаштувань Gulp.js:

```
var gulp = require('gulp'), //підключаєм gulp
   sass = require('gulp-sass'), // підключаєм SASS пакет
   autoprefixer = require('gulp-autoprefixer'); //пагін
   autoprefixer
var config = {
   sassPath: 'www/sass' // шлях до SASS файлів
};
var styles =['/page1.scss','/page2.scss']; //SASS файли
для компіляції
var src=[];
for(var i in styles){
src.push(config.sassPath + styles[i]);}
//таксk 'css' для компіляції
gulp.task('css', function() {
   return gulp.src(src)
      .pipe(sass({
         style: 'compressed',
         loadPath: [
            config.sassPath,
            config.sassPath+'/assets/stylesheets'
         ]
      }))
      .pipe(autoprefixer({
         browsers: ['last 2 versions','ie 9']
      }))
      .pipe(gulp.dest('www/css'))
});

   gulp.task('watch', 'css', function() {
gulp.watch(config.sassPath + '/*.scss', ['css']);
});

gulp.task('default', ['watch']);
```

Важливим аспектом в створенні веб сторінок є можливість їхнього відображення за допомого різних веб браузерів. Одним із способів досягнення крос-браузерності є плагін, написаний для Gulp.js, autoprefixer. Робота плагіну полягає в тому, що він дописує необхідні префікси до CSS властивостей і при необхідності модифікує їх, таким чином забезпечуючи крос-браузерність веб сторінки.

Опишемо роботу плагіну autoprefixer на прикладі CSS властивості «linear-gradient». «Linear-gradient» — дозволяє задавати градієнтний фон елементу. Нижче наведені приклади звичайного і крос-браузерного веб елемента:

1. Елемент, який не підтримує крос-браузерності:

```
div{background: linear-gradient(to bottom, #444444,
#999999);}
```

Вище наведений приклад не буде працювати у всіх веб браузерах.

2. Крос-браузерний елемент:

```
div{
    background: -webkit-gradient(linear, left top, left bottom, from(#444444), to(#999999)); /* Saf4+, Chrome */
    background: -webkit-linear-gradient(top, #444444, #999999); /* Chrome 10+, Saf5.1+, iOS5+ */
    background: -moz-linear-gradient(top, #444444, #999999); /* FF3.6+ */
    background: -ms-linear-gradient(top, #444444, #999999); /* IE10 */
    background: -o-linear-gradient(top, #444444, #999999); /* Opera 11.10+ */
    background: linear-gradient(to bottom, #444444, #999999);
}
```

Можна побачити, що код, наведений в другому прикладі, більший за обсягом і менш зрозумілий. Для того, щоб уникнути таких незручностей використовується плагін autoprefixer. Не має потреби вручну дописувати префікси до CSS властивостей. До того ж, запам'ятати які саме властивості вимагають наявності префіксу чи іншого написання просто неможливо.

У висновку можна сказати що, потрібно використовувати технології, які покращують розробку, організовують код і економлять час на написання і доопрацювання проектів. Звичайно, необхідно затратити деякий час, для того, щоб вивчити нову технологію і зрозуміти принцип її роботи. Але ці знання дозволяють продуктивно працювати і саморозвиватися.

Література

1. CSS with superpowers [Електронний ресурс] / Сайт SASS. Режим доступу: http://sass-lang.com/, вільний. — Загл. з екрану.

2. Gulp API docs [Електронний ресурс] / Режим доступу: https://github.com/gulpjs/gulp/blob/master/docs/API.md /, вільний. — Загл. з екрану.