Науковий керівник – асист. Луцик І. Б.

ВЕРСТКА WEB-СТОРІНОК ЗА ДОПОМОГОЮ ІНСТРУМЕНТА GULPJS

Постановка проблеми. Одним із основних засобів комунікацій та спілкування між людьми як у бізнесі, так і в побуті став Інтернет. Для того, щоб привернути увагу відвідувача, а також для зручності наповнення сайту, необхідно використовувати найновіші технології. Тому актуальним є вивчення сучасних способів верстки web-сторінок із застосуванням наборів різноманітних бібліотек із великим, добре розширюваним функціоналом інструментів для швидкого виконання рутинних завдань.

Найчастіше верстка сторінок сайту здійснюється на основі заздалегідь створеного графічного макету. Від якості виконаної роботи залежить відповідність отриманого результату вихідних файлів, а також сприйняття ресурсу як з боку відвідувачів, так і з боку браузерів і пошукових роботів.

Метою статті є представлення способу верстки web-сторінок за допомогою інструмента GulpJS, який забезпечує можливість динамічного створення сторінок.

Основна частина. Якісна робота повинна відповідати певним стандартам і необхідно підібрати оптимальне рішення в поєднанні зі втіленням інноваційних ідей і методів.

Розглянемо налаштування середовища для здійснення верстки з допомогою таск-менеджера GulpJS.

Таск-менеджер – невеликий додаток, який використовується для автоматизації нудних і рутинних, але від того не менш важливих завдань, які доводиться постійно виконувати в процесі розробки проекту. Такі завдання включають в себе конкатенацію, компіляцію файлів, мініфікацію, препроцессінг CSS. Досить проста ідея, яка дозволяє заощадити дуже багато часу і допомагає зберігати фокус на завданнях, пов'язаних безпосередньо з розробкою проекту.

Виконувати усі завдання ми будемо у середовищі «Ubuntu 16.04», засобами текстового редактора «Sublime». Для коректної робити менеджера, необхідно встановити бібліотеки «Node.JS» та «Npm». Їх можна у вільному доступі знайти на офіційних сторінках(www.npmjs.com, nodejs.org).

Спочатку формуємо структуру каталогів проекту. Зазвичай, потрібно не менше, ніж 2 папки. Одна (src) в якій ми власне писатимемо код, і друга (build), в яку складальник буде записувати готові файли. У домашньому каталозі створюємо файл – «gulpfile.js». Це наш основний таск-файл. У ньому ми будемо додавати всі функції, які повинен виконувати наш таск-менеджер. Для встановлення GulpJS, виконуємо наступну команду у терміналі:

$ sudo npm install gulp –g та *$ npm init*

Після цього вводимо дані, які запитує програма. Натискаємо «Enter» і наш проект готовий до використання. Для того, щоб почати верстку, необхідно встановити плагіни.

Менеджер має велику бібліотеку плагінів, які можна переглянути на сайті gulpjs.com. Ми встановлюємо наступні:

gulp-jade – потужний шаблонізатор з мінімалістичним синтаксисом;

gulp-sass – sass (англ. Syntactically Awesome Stylesheets) — скриптова метамова, яка інтерпретується в каскадні таблиці стилів (CSS);

gulp-watch – система реагування на зміни файлів;

gulp-html-prettify – форматує html-файли, зручні для читання;

browser-sync – дає можливість запустити сервер, на якому можна виконувати свої додатки, піклується про перезавантаження HTML/PHP-файлів.

Для завантаження плагінів, виконуємо наступну команду:

$ npm install gulp gulp-jade gulp-sass gulp-watch gulp-html-prettify browser-sync –save-dev

Потрібні плагіни встановлено, отож приступаємо до написання завдань для нашого таск-менеджера. Завдання прописуються у файлі «gulpfile,js».

Спочатку підключаємо всі встановлені плагіни:

```
var gulp = require('gulp'),
  jade = require('gulp-jade'),
  sass = require('gulp-sass'),
  watch = require('gulp-watch'),
  prettify = require('gulp-html-prettify'),
  browserSync = require('browser-sync').create();
```

Далі, додамо завдання для створення «html» файлів із формату «jade»:

```
gulp.task('jade', function() {
   return gulp.src('app/templates/*.jade')
   .pipe(jade())
   .pipe(prettify({
     unformatted: []
   }))
```

```
    .pipe(gulp.dest('app/builds/'));
});
```

У даному завданні, після виконання функції ми вказали директорію де знаходяться файли «.jade» (gulp.src('app/templates/*.jade')), які мають бути компільовані у формат «.html». Pipe(jade()) – компілює, .pipe(prettify({unformatted: []})) - форматує код у ієрархічній структурі html(рис.1).

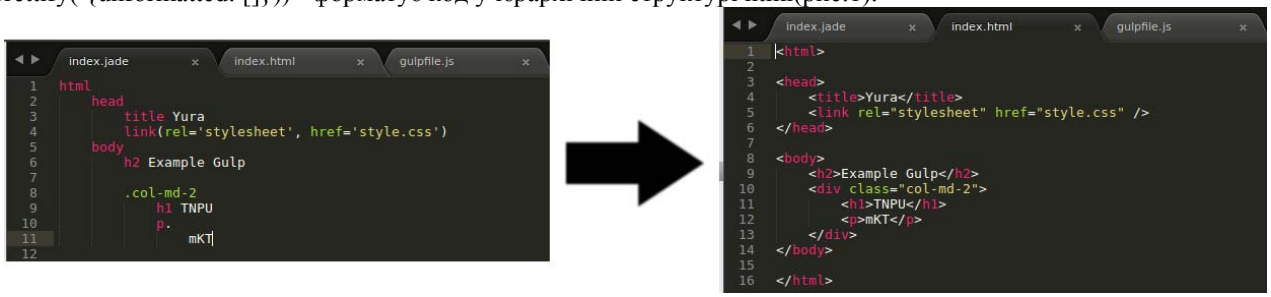

Рис.1. Компіляція із формату «Jade» у «Html»

Аналогічно до даного прикладу, формуємо завдання для файлів стилів, які пишуться у форматі «.sass», що також спрощує роботу верстальника, а потім компілюються у формат «.css», який використовується для веб-сторінок:

```
gulp.task('sass', function(){
  return gulp.src('app/scss/*.scss')
   .pipe(sass().on('error', sass.logError))
   .pipe(gulp.dest('app/builds'))
});
```

Виконуючи команди «$ gulp sass» та «$ gulp jade» наш менеджер буде форматувати та компілювати файли, але кожного разу запускати команди при внесенні змін, не зручно. Тому ми використовуємо плагін gulp-watch, який повинен слухати зміни і при кожному збереженні файлів автоматично запускати завдання для форматування та компіляції..

Напишемо завдання для слухача:

```
gulp.task('watch', ['jade', 'sass'], function () {
  gulp.watch('app/templates/*.jade', ['jade']);
  gulp.watch('app/scss/*.scss', ['sass']);
});
```

Функція завдання “watch” буде слідкувати за змінами, які вносяться у файли вказаних директорій із заданим розширенням(app/templates/*.jade'), та виконувати відповідні завдання['jade']. Ми можемо відслідковувати всі зміни у терміналі, не запускаючи кожного разу завдань, вони автоматично запускатимуться при внесенні та збереженні змін(рис.2).

Рис.2. Автоматичний запуск завдань при внесенні змін

Для того, щоб не перезавантажувати веб-сторінку кожного разу при внесенні змін, використовуємо плагін «browser-sync». Він дозволяє запускати локальний сервер, та автоматично перезавантажувати сторінку, коли вносяться зміни. Додаємо наш локальний сервер до завдання «watch», тепер наша функція буде починатися із наступного коду:

```
browserSync.init({
    server: {
       baseDir: "app/builds/"
    }
```

});

Додаємо команди до завдань, при виконанні яких ми хочемо перезавантажувати сторінку:

.pipe(browserSync.stream());

Наш таск-файл повністю готовий до використання. Запускаємо команду «$ gulp watch» і виконуємо верстку сторінки набагато швидше і якісніше, та динамічно спостерігаємо які зміни відбуваються на нашій веб-сторінці.

Додаємо зміни у файл «index.jade», та слідкуємо за змінами(рис.3):

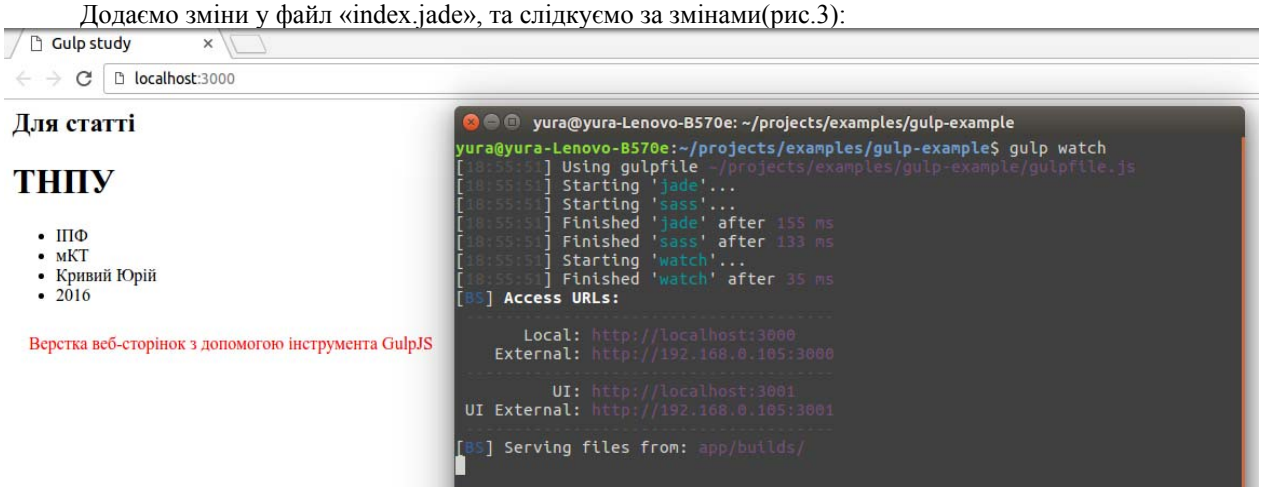

Рис.3. Робота створеного сервера

Таким чином, для повноцінного створення сучасного веб-сайту необхідними є вміння працювати із таск-менеджерами та їх плагінами.

Розроблений на основі використання інструмента GulpJS проект розміщений у мережі «Github» за адресою: «https://github.com/georgebent/gulp-example».

Висновки. Для динамічної верстки веб-сторінок доцільним є використання інструмента GulpJS, який використовується для автоматизації нудних і рутинних, але від того не менш важливих завдань, які доводиться постійно виконувати в процесі розробки проекту. Такі завдання включають в себе конкатенацію, компіляцію файлів, мініфікацію, препроцессінг CSS. Це дозволяє заощадити дуже багато часу, і допомагає зберігати фокус на завданнях, пов'язаних безпосередньо з розробкою проекту.

ЛІТЕРАТУРА

1. Maynard T. Getting Started with Gulp / Travis Maynard. – Birmingham, 2015. – 120 с. – (Packt Publishing Ltd.).
2. Матеріали офіційного сайту таск-менеджера GulpJS : [Електронний ресурс]. Режим доступу: http://gulpjs.com/

Злонкевич Р.

Науковий керівник – викл. Сіткар Т. В.

ПРОЕКТУВАННЯ МОБІЛЬНОГО ДОДАТКУ ДЛЯ ОРГАНІЗАЦІЇ НАВЧАЛЬНОЇ ДІЯЛЬНОСТІ «ОРГАНАЙЗЕР ВЧИТЕЛЯ»

Актуальність проблеми. Діяльність викладача здійснюється на підставі Статуту закладу, Концепції виховної роботи з огляду на особливості й традиції закладу. Зміст його діяльності визначається Законом України «Про вищу освіту», Державною національною програмою «Освіта» («Україна XXI століття»), «Концепцією виховання дітей та молоді у національній системі освіти», «Національною доктриною розвитку освіти України у XXI столітті», відповідними інструктивно-методичними документами Міністерства освіти і науки України, а також положеннями, розробленими структурними ланками[1].

У роботі викладача передбачається виховна діяльність академічної групи, створення умов для самовираження кожного студента і розвитку кожної особистості. Доступ викладача до додатку який відображає успішність кожного студента у групах де він викладає, дозволить йому стимулювати їх до навчання та контролювати їхню відвідуваність, а також при необхідності взаємодіяти з батьками студента.

Тому для підвищення ефективності роботи викладача, потрібно розробити таке ПЗ, як дозволить викладачу контролювати успішність та відвідування студентами його пар, та в разі потреби корегувати його за допомогою його прямого спілкування з студентом окремо, та в разі необхідності і з його батьками.