Web Advanced: Javascript APIs

“We will learn JavaScript properly. Then, we will learn useful design patterns. Then we will pick up useful tools for making cool things better.”

FALL 2018

SESSION #10

INTRODUCTION TO DEVELOPMENT WORKFLOWS

jaink@newschool.edu

https://canvas.newschool.edu/courses/1407281

https://classroom.github.com/classrooms/42809645-parsons-web-advanced-javascript-fall-2018

RECAP

WHAT IS A WORKFLOW?

- ➔ Organize the js and scss, css, assets
- ➔ better integration with source control
- ➔ Automate repetitive tasks like joining, minifying, parsing SASS, moving and renaming files etc.
- ➔ allow easy replication on other environments/team systems without changing the source code
- ➔ No more FTP!!!

COMPONENTS OF A WORKFLOW

- ➔ Source Control: Git
- ➔ Allows managing code changes over time, along with actions like alternative copies (branches), reverting the code to previous states (commits) whenever needed etc.
- ➔ Also allows better code management when working with teams in parallel.
- ➔ Github - a service used for hosting git repositories (free for open source projects)
- ➔
- ➔ Easy guide here: http://rogerdudler.github.io/git-guide/

COMPONENTS OF A WORKFLOW

➔ JS Transpiler: Babel/Typescript

➔ Required to convert modern/edge code like ES6, Typescript etc. for all browsers.
➔ Required to convert the language into ES5.
➔ Eventually ES6 will be 1005 supported and this component will not be necessary if all code is written in ES6 directly.
➔ Babel is still handy to completely future proof the code.

COMPONENTS OF A WORKFLOW

- Task Runner: GRUNT/GULP

- Runs automated tasks on code to generate a cleaner/optimized output
- Handle all repetitive tasks, manages all the heavy lifting

COMPONENTS OF A WORKFLOW

- CSS Preprocessors: LESS/SASS

- SCSS is a scripting language that extends CSS that eventually flattens/compiles into regular CSS.
- Allows for more programmatic approaches to writing CSS styles.
- Allows features like reusable variables, nested definitions, importable modules, mixins/functions etc.

COMPONENTS OF A WORKFLOW

- ➔ Code Linting: JSLINT/ESLint

- ➔ Linting checks for bugs or inconsistency in code before compiling or processing.
- ➔ Issues can be simple typos, missing punctuation etc. and most lint systems allow a customizable definition of standards to test the code against, in real time.

COMPONENTS OF A WORKFLOW

- Integrated Testing: Mocha/Jasmine/Selenium

- Requires writing specific code for each functionality in the application that tests all possible conditions.
- These tests are then run through the framework used and produces results, without manually debugging/logging etc.
- Requires time/patience and experience to write clean and comprehensive tests.
- Unit tests small pieces of code like functions, aloing and isolated to verify the cleanliness of data going in and out.
- Integration tests overall system integration and needs proper scripting.
- Function tests performs actual browser and UI testing.

REQUIREMENTS

- ➔ A little familiarity with the Terminal
- ➔ Xcode (OSX)
- ➔ Homebrew (OSX)
- ➔ NPM

SETUP

Xcode:

```
https://developer.apple.com/download
```

```
gcc -v
```

```
xcode-select --install
```

Homebrew: package manager for OSX

```
/usr/bin/ruby -e "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/instal
l/master/install)"
```

```
brew update
```

```
brew doctor
```

Add the brew location in the profile file:

```
export PATH="/usr/local/bin:$PATH"
echo 'export PATH="/usr/local/sbin:$PATH"' >>
~/.bash_profile
```

Install Nodejs (also installs NPM):

```
brew install node
```

NODE PACKAGE MANAGER

- ➔ NPM: package manager for javascript package libraries
- ➔ Installed with Node
- ➔ Contains a massive number of libraries of reusable code for Node and other javascript based applications
- ➔ https://www.npmjs.com/

Current version: 9.11.1

```
node -v
```

To update to latest Node:

```
npm install npm@latest -g
```

GIT INSTALLATION

- ➔ Git will track all versions and changes to the code
- ➔ https://git-scm.com

```
brew install git
```

Create a new repo:

```
git init
```

Or clone an existing one:

```
git clone username@host:/path/to/repository
./project_folder
```

Typical commands:

```
git add *

git commit -m "Commit message"

git checkout master

git checkout -b feature_x
```

GUI (OSX): https://www.sourcetreeapp.com/

GULP INSTALLATION

- **Gulp is a task runner to handle common and frequently run tasks to automate it through a script and plugins. eg.**
- Lint JS and CSS
- Minify CSS and JS
- Autoprefix CSS
- SASS, LESS Compilation
- Minify Images
- Auto Generated SVG Sprites
- Build production ready files with file size reporting
- Uglify JS and CSS for production and my favourite,
- BrowserSync
- https://www.npmjs.com/

To install globally:

```
npm install --global gulp gulp-cli
```

PROJECT INITIALIZATION

- ➔ Each project needs a config file called package.json that will record all the package dependencies needed for the tasks.
- ➔ All packages installed "LOCALLY" will get added to this file.
- ➔ All dependencies get downloaded into a folder inside this project ready to be used.

In Terminal go to the project folder:

```
cd "~/Documents/D&T/Faculty 2018/class 10"

npm init

npm install --save-dev gulp
```

Install some commonly used plugins:

```
npm install --save-dev gulp-sass gulp-cssnano
gulp-sourcemaps gulp-autoprefixer
```

SETUP THE TASKRUNNER

➔ **Create gulpfile.js**

```
'use strict';

var gulp = require('gulp');
var sass = require('gulp-sass');
var cssnano = require('gulp-cssnano');
var sourcemaps = require('gulp-sourcemaps');
var autoprefixer = require('gulp-autoprefixer');

gulp.task('workflow', function () {
  gulp.src('./src/sass/**/*.scss')
    .pipe(sourcemaps.init())
    .pipe(sass().on('error', sass.logError))
    .pipe(autoprefixer({
      browsers: ['last 2 versions'],
      cascade: false
    }))
    .pipe(cssnano())
    .pipe(sourcemaps.write('./'))

  .pipe(gulp.dest('./dist/css/'))
});

gulp.task('default', function () {
  gulp.watch('./src/sass/**/*.scss', ['workflow']);
});
```

JS TASKRUNNER

```
$ npm install jshint gulp-jshint gulp-concat gulp-uglify
gulp-rename --save-dev
```

➔ **In gulpfile.js:**

```
var jshint = require('gulp-jshint');
var concat = require('gulp-concat');
var uglify = require('gulp-uglify');
var rename = require('gulp-rename');

gulp.task('lint', function() {
    return gulp.src('src/js/*.js')
        .pipe(jshint())
        .pipe(jshint.reporter('default'));
});

// Concatenate & Minify JS
gulp.task('scripts', function() {
    return gulp.src('./src/js/*.js')
        .pipe(concat('scripts.js'))
        .pipe(gulp.dest('./dist/js'))
        .pipe(rename('scripts.min.js'))
        .pipe(uglify())
        .pipe(gulp.dest('./dist/js'));
});

gulp.task('default', function() {
    gulp.watch('./src/js/*.js', ['lint', 'scripts']);
    gulp.watch('./src/sass/**/*.scss',
['sassworkflow']);
})
```

AUTO LOAD BROWSER

```
$ npm install --save-dev browser-sync

var browserSync = require('browser-sync').create();

gulp.task('browserSync', function() {
    browserSync.init({
        server: {
            baseDir: './',
            index: "index_empty.html"
       },
    })
})

gulp.task('sassworkflow', function () {
    gulp.src('./src/sass/**/*.scss')
    // tasks go here
    .pipe(sourcemaps.init())
    .pipe(sass().on('error', sass.logError))
    .pipe(autoprefixer({
        browsers: ['last 2 versions'],
        cascade: false
    }))
    .pipe(cssnano())
    .pipe(sourcemaps.write('./'))
    .pipe(gulp.dest('./dist/css/'))
    .pipe(browserSync.reload({
      stream: true
    }));
});
```

EXAMPLES

Assignment

Next Steps

Plugins and Modules