Workflows Project

This is a workflow test project

Some setup tips

Note:To Show Hidden files in Finder on Mac use the follwoing command in the terminal

```
defaults write com.apple.finder AppleShowAllFiles YES
```

Install GIT on your System

```
git —version to check if installed
```

Install Node

```
Node (node —v to check if installed)
```

Install Brew

This package lets you install packages directly to your mac os

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

Install Imagemagick and GraphicsMagick using Brew

I went to Apple Developer website and downloaded the Command Line Tools dmg directly because I was getting an error with the following commands. The Command Line Tools can be found under - https://developer.apple.com/download/more/
(https://developer.apple.com/download/more/)

```
brew install imagemagick
brew install graphicsmagick
```

On Windows you could use Scoop instead of Brew

```
scoop install sudo (In case you haven't got it already)
sudo scoop install imagemagick -g
```

Other Global Installs

Install these to get started if starting from fresh. If you are downloading this folder

```
GULP - Npm install –g gulp
BROWSERIFY - Npm install –g browserify
SASS - Gem install sass
COMPASS - Gem install compass
RUBY - Install ruby if on PC
```

Also create Github account and keep your details

Create folder structure and Files for Project

`Ls` to list current files in folder (ls –a to show hidden files)

`Cd` to change directory

`Clear` to clear screen

```
- workflows
  - builds
    - development
      - css
      - images
      - js
      - index.html
    - production
  - components
    - coffee
    - sass
    - scripts
```

Setup your package.json file

```
npm init
```

Note: Make sure to do this command from the terminal and with your project folder selected

Setup your GIT

Create a .gitignore file and place the following code in it

```
.DS_Store
node_modules
.tmp
.sass-cache
builds/**/images/*
```

These are the files you dont want to keep track off on git. This file should be in the root of your project folder

From the project folder type:

```
git init
git status
git commit -m "First Commit"
git log
```

To exit out of your git log press 'q' to quit

Now to push your files to the online repository

```
git remote add origin https://github.com/montaynej/workflows.git
git push -u origin master
```

You may be asked to enter your username and password for first usage

If you require to return to a version of your git do the following

```
git reset --hard 71c27777543ccfcb0376dcdd8f6777df055ef479
```

Using the code for your commit instead, then

```
git push --force
```

To update on gitHub

Setting up Gulp within your project

Type the two following commands to install Gulp and Gulp Util in your project directory

```
npm install —save-dev gulp
npm install —save-dev gulp-util
```

Notice how the package.json file was edited

We need to create a gulpfile.js to tell it what we want it to do. In the gulpfile.js type

```
var gulp = require('gulp');
var gutil = require('gulp-util');

gulp.task('log', function () {
    gutil.log('Workflows are awesome');
});
```

Then type `gulp log` in the command line to see if it is working Now follow commands to

commit and push the changes to GIThub

Setting up Gulp to process Coffee Script

Ok so lets do some processing of the coffee script. We need Gulp-coffee so lets install that

```
npm install -save-dev gulp-coffee
```

Add this to gulpfile.js and test

```
Var coffee = require('gulp-coffee')
var coffeeSources = ['components/coffee/tagline.coffee'];
gulp.task('coffee', function () {
    gulp.src(coffeeSources)
        .pipe(coffee({
                bare: true
            })
            .on('error', gutil.log))
        .pipe(gulp.dest('components/scripts'))
});
```

Setting up Gulp Concat

Install Gulp concat to use this utility

```
Npm install –save-dev gulp-concat
```

Add this to gulpfile.js

```
Var concat = require('gulp-concat')
var jsSources = ['components/scripts/rclick.js', 'components/scripts/pixgrid.js',
'components/scripts/tagline.js', 'components/scripts/template.js'
];
gulp.task('js', function () {
    gulp.src(jsSources)
        .pipe(concat('script.js'))
        .pipe(gulp.dest('builds/development/js'))
});
```

Install Browserify

Now we need to install browserify which allows us to install our libraries as dependendcies

```
Npm install –save-dev gulp-browserify
```

Now install the jquery library and Mustache

```
Npm install –save-dev jquery
```

```
Npm install –save-dev mustache
```

Now Add the following to gulpfile.js

```
Var browserify = require('gulp-browserify')
```

```
And edit the gulp JS task to the following
gulp.task('js', function () {
    gulp.src(jsSources)
        .pipe(concat('script.js'))
        .pipe(browserify())
        .pipe(gulp.dest('builds/development/js'))
});
```

Anywhere within the jsSources that has a require browserify will implement the required script

In the tagline.coffee file add the following line

```
$ = require 'jquery' on the top
```

Then test with gulp coffee and then gulp js.

Installing and working with Sass file

Install gulp compass using

```
Npm install –save-dev gulp-compass
```

And add it to the gulpfile.js

```
Var compass = require('gulp-compass')
```

Add a variable for sass sources using the following

```
var sassSources = ['components/sass/style.scss'];
```

Remember sass has it own import command so we don't need to add all sources

```
gulp.task('compass', function () {
    return gulp.src(sassSources)
        .pipe(compass({
            css: 'builds/development/css',
            sass: 'components/sass',
            image: 'builds/development/images',
            comments: 'true',
            style: 'expanded'
        }))
```

```
        .on('error', gutil.log)

    .pipe(gulp.dest('builds/development/css'));

});
```

Issue task and watch Sequences

if you want a task to run but you need another task to run before it then use the example below

```
gulp.task('js', ['coffee'] function () {
    gulp.src(jsSources)
        .pipe(concat('script.js'))
        .pipe(browserify())
        .pipe(gulp.dest('builds/development/js'))
});
```

We dont need this in our project so make sure to revert to original

To run tasks in Sequence you can create a new task and pass it the information of the task sequence

```
gulp.task('all',['coffee','js','compass']);
```

We should name this task as default and when we type gulp in the terminal window it will run the default task

```
gulp.task('default',['coffee','js','compass']);
```

To add a watch task we get it to keep an eye on our coffeeSources (which is an array of files) and if any of those files get changed or updated we instruct it to execute the coffee task

```
gulp.task('watch', function () {
    gulp.watch(coffeeSources, ['coffee'])
});
```

When you issue the gulp watch task notice how the task does not complete and return you to the terminal. Change the tagline.coffee file and notice how it changes the tagline.js file in scripts. It doesnt change to outputted script in the development js folder we need to also add a watch to the js folder.

```
gulp.task('watch', function () {
    gulp.watch(coffeeSources, ['coffee']);
    gulp.watch(jsSources,['js'])
});
```

To terminate the watch function in the terminal window press `ctrl C` and run the gulp watch command again

No lets add the gulp watch task to look at sass changes.Remember the variable sassSources only refers to the style.scss file so we need to tell it to watch all of the files in the sass folder. We do this by adding the following

```
gulp.watch('components/sass/*.scss', ['compass']);
```

You should also add the watch task to the end of the default task so when the project starts the watch command will be executed.

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

Adding LiveReload and Live Server

This package allows you to run a live server and also have live preview. Firstly install gulp connect.

```
npm install --save-dev gulp-connect
```

Add this to your gulpfile.js

```
var connect = require('gulp-connect')
```

Add the follwoing task function to your gulpfile.js

```
gulp.task('connect', function () {
    connect.server({
        root: 'builds/development/',
        livereload: true
    });
})
```

Also add this task to the default task

```
gulp.task('default', ['coffee', 'js', 'compass', 'connect','watch']);
```

Now add .pipe(connect.reload()) to end of your js task and your compass task. There is no need to add it to the coffee script task and this generate a js file which will in turn connect.reload

```
gulp.task('js', function () {
    gulp.src(jsSources)
        .pipe(concat('script.js'))
        .pipe(browserify())
        .pipe(gulp.dest('builds/development/js'))
        .pipe(connect.reload())
});
```

```
gulp.task('compass', function () {
    return gulp.src(sassSources)
        .pipe(compass({
            css: 'builds/development/css',
            sass: 'components/sass',
            image: 'builds/development/images',
            comments: 'true',
            style: 'expanded'
        }))
        .on('error', gutil.log)
        .pipe(gulp.dest('builds/development/css'))
        .pipe(connect.reload());
});
```

Now entry this address into your address bar in Chrome and test.

```
http://localhost:8080/
```

You Could also write a gulp-connect stopping task as follows

```
gulp.task('stopServer', function() {
  connect.server({
    port: 8888
  });

  connect.serverClose();
});
``

Add the follwoing to your style.scss under the compass import and notice how the p
age margins are reset in the live server.
```

@import "compass/reset";

```
## Adding Static Reloads for .html & .json files
Fitstly create a variable for the HTML sources
```js
var htmlSources = ['builds/development/*.html ']
```

Then create a task for html in the gulpfile.js

```
gulp.task('html', function () {
    gulp.src(htmlSources)
        .pipe(connect.reload())
});
```

Add it to the default task and the watch task

```
gulp.task('default', ['html', 'coffee', 'js', 'compass', 'connect', 'watch']);
```
```

```
gulp.task('watch', function () {
    gulp.watch(coffeeSources, ['coffee']);
    gulp.watch(jsSources, ['js']);
    gulp.watch('components/sass/*.scss', ['compass']);
    gulp.watch(htmlSources, ['html'])
});
```

We need to also install a task to watch for changes in the json file.

```
gulp.task('json',function(){
    gulp.src('builds/development/js/*.json')
    .pipe(connect.reload())
})
```

Add it to our default task and our watch task

```
gulp.task('default', ['html', 'json', 'coffee', 'js', 'compass', 'connect', 'watch'
]);
```

```
gulp.watch('builds/development/js/*.json', ['json'])
```

Now edit your .json file and save. Watch it reload with new content.

Setting up Environment Variables

We start by setting up an environment variable. This is added at the top of the gulpfile.js. This ets up the ENV variable if it hasnt been set through the terminal window. We will change it through the terminal window later.

```
var env = process.env.NODE_ENV || 'development';
```

Set up your variable in the gulpfile.js so that your declarations and assignments are seperated.

```
var env,
    coffeeSources,
    jsSources,
    sassSources,
    htmlSources,
    jsonSources,
    outputDir;

env = process.env.NODE_ENV || 'development';
coffeeSources = ['components/coffee/tagline.coffee'];
jsSources = [
    'components/scripts/rclick.js',
    'components/scripts/pixgrid.js',
    'components/scripts/tagline.js',
    'components/scripts/template.js'
];
```

```
sassSources = ['components/sass/style.scss'];
htmlSources = ['builds/development/*.html'];
jsonSources = ['builds/development/js/*.json'];
```

Now add a conditional statement after the env declaration

```
if (env === 'development') {
    outputDir = 'builds/development/';
}
else {
    outputDir = 'builds/production/'
}
```

Now where every you used 'builds/development' within your gulpfile.js you need to replace it 'outputDir'. For example

```
htmlSources = [outputDir + '*.html'];
```

Now to change the environment variable to production we type the following into the terminal window and execute the gulp command after it. Watch for the spaces!!!

```
NODE_ENV=production gulp
```

Add sassStyle to the variable list and then add this conditional statement

```
outputDir,
sassStyle;
```

```
if (env === 'development') {
    outputDir = 'builds/development/';
    sassStyle = 'expanded'
}
else {
    gutil.log('Workflows are yawesome');
    outputDir = 'builds/production/';
    sassStyle = 'compressed'
}
```

Now edit the compass style to use that variable

```
gulp.task('compass', function () {
    return gulp.src(sassSources)
        .pipe(compass({
            css: outputDir + 'css',
            sass: 'components/sass',
            image: outputDir + 'images',
            comments: 'true',
            style: sassStyle
        }))
        .on('error', gutil.log)
```

```
        .pipe(gulp.dest(outputDir + 'css'))
        .pipe(connect.reload());
});
```

Adding Gulp If - Conditional Variables

Gulp If allows you to put conditional statements within the gulp piped flow. Firstly lets install it

```
npm install --save-dev gulp-if
```

Add it to the list of requires at the top of the gulpfile.js

```
gulpIf = require('gulp-if');
```

Now lets try to Uglify our Javascript using the gulp-uglify plugin. Firstly lets install it

```
gulpif = require('gulp-uglify');
```

Add it to the list of requires at the top of the gulpfile.js

```
uglify = require('gulp-uglify');
```

Now add the condition in stream to the js task

```
gulp.task('js', function () {
    gulp.src(jsSources)
        .pipe(concat('script.js'))
        .pipe(browserify())
        .pipe(gulpIf(env === 'production', uglify()))
        .pipe(gulp.dest(outputDir + '/js'))
        .pipe(connect.reload())
});
```

Run NODE_ENV=production gulp from the Command line and test

```
NODE_ENV=production gulp
```

Copying and minifying HTML from Development to Production

We need to use gulp-minify-html. Install this

```
npm install --save-dev gulp-minify-html
```

In the gulpfile.js add a require to the top of the document for this plugin

```
minifyHTML =require('gulp-minify-html')
```

In the HTML task add the following code. Make sure to change the gulp.src value as we will only by watching and changing the one html file taht sits in the development folder

```
gulp.task('html', function () {
    gulp.src('builds/development/*.html')
        .pipe(gulpIf(env === 'production', minifyHTML()))
        .pipe(gulpIf(env === 'production', gulp.dest(outputDir)))
        .pipe(connect.reload())
});
```

We need to modify the watch task also by changing the htmlSources to watch the html in the development folder only

```
gulp.task('watch', function () {
    gulp.watch(coffeeSources, ['coffee']);
    gulp.watch(jsSources, ['js']);
    gulp.watch('components/sass/*.scss', ['compass']);
    gulp.watch('builds/development/*.html', ['html']);
    gulp.watch(outputDir + 'js/*.json', ['json'])
});
```

Copying and minfying our Json file from Development to Production

Firstly install the gulp-jsonminify plugin

```
npm install --save-dev gulp-jsonminify
```

We also need to add a require at the top of the document

```
jsonMinify =require('gulp-jsonminify')
```

Edit the json task to the following

```
gulp.task('json', function () {
    gulp.src('builds/development/js/*.json')
        .pipe(gulpIf(env === 'production', jsonMinify()))
        .pipe(gulpIf(env === 'production', gulp.dest('builds/production/js')))
        .pipe(connect.reload())
});
```

We need to modify the watch task also by changing the json to watch the json in the development folder only

```
gulp.task('watch', function () {
    gulp.watch(coffeeSources, ['coffee']);
    gulp.watch(jsSources, ['js']);
    gulp.watch('components/sass/*.scss', ['compass']);
    gulp.watch('builds/development/*.html', ['html']);
    gulp.watch('builds/development/js/*.json', ['json'])
});
```

Compressing Images

We need to use 2 plugins for this part. We install gulp-imagemin and add imagemin-pngcrush to our requires

```
npm install --save-dev gulp-imagemin
npm install --save-dev imagemin-pngcrush
```

We also as before need to add these plugins to the require

```
imageMin = require('gulp-imagemin'),
pngCrush = require('imagemin-pngcrush'),
```

We now need to create an images task

```
gulp.task('images', function () {
    gulp.src('builds/development/**/*.*')
        .pipe(gulpIf(env === 'production', imageMin({
            progressive: true,
            scgoPlugins: [{
                removeViewBox: false
            }],
            use: [pngCrush()]
        })))
        .pipe(gulpIf(env === 'production', gulp.dest(outputDir + 'images')))
        .pipe(connect.reload())
})
```

We need to edit out default to taks to add the images task to it

```
```
We need to edit our watch task to look out for image changes
```js
```

Installing Image Resize

This package requires that you have imageMagick and graphicsmagick installed using Brew. See above. Type the following into the terminal window

```
npm install --save-dev gulp-image-resize
```

Add the following to your gulpfile.js header

```
var imageResize = require('gulp-image-resize');
```

And also add the following into your gulpfile.js

```
gulp.task('image', function () {
  gulp.src('test.png')
    .pipe(imageResize({
      width : 100,
      height : 100,
      crop : true,
      upscale : false
    }))
    .pipe(gulp.dest('dist'));
});
```

Place image in folder and define destination and then test