АЛГОРИТМ формирования СБОРКИ
(для front -end проектирования)
(на основе Учебной сборки от -Loftschool)

1. Создаем папки проекта через терминал - mkdir css
2. Создаем файл - touch css/main.css
3. В PyCharm - открываем новый проект
3. Инициализируем новый проект: - npm int -y (yarn init -y)
при этом создается файл - package.json
4. Инициалмзируем - Git — git int
5. В папке открываем скрытые файлы - Cmd + Shift + .
6. В терминале создаем файл — touch .gitignore
7. Открываем его в PyCharm и редактируем
8. Для работы с GULP - создаем файл - touch gulpfile.js
9. Инсталируем GULP 4.0 - npm install -D gulpjs/gulp#4.0 (yarn add gulpjs/gulp#4.0)
Это добавляет в проект папку - node_modules и в фай package.json - устанавливает зависимость - gulp
10. Устанавливаем - csso — npm i -D gulp-csso (yarn add gulp-csso) - появиься еще одна зависимость в package.json (Эта утилита убирает дублирующий код в css)
11. Проверяем работу GULP и CSSO - в файл main.css - создаем пример дубля:

```
body {
  background: grey;
}
body {
  color: red;
}
```

12. В gulpfile.js - прописываем 1-ое задание:

```
var gulp = require ('gulp');
var csso = require ('gulp-csso');

gulp.task('default', function () {
    return gulp.src('css/*.css')
        .pipe(csso())
        .pipe(gulp.dest('build/css'));
});
```

12. Запускаем на выполнение - в случае с yarn: gulp
В случае с npm - сначала добавляем в package.json

```
"scripts": {
    "serg": "gulp",
    "test": "echo \"Error: no test specified\" && exit 1"
  },
```

затем запускаем - npm run serg
Так же эти строки можем добавлять и в случае с yarn и тогда запустеем по команде - yarn serg
13. Чтобы избежать добавления этого скрипта - глобально командный интерфейс для GULP
npm install -g gulp-cli (yarn add -g gulp-cli)

14. Удаляем папку build - rm -r build css
15. Создаем две папки рекурсивно - mkdir -p source/style
16. Создаем файл - app.scss - touch source/style/app.scss - заполняем файл (если нужно)
17. Устанавливаем пакеты : npm i -D (yarn add)
gulp-load-plugins gulp-sass gulp-sourcemaps gulp-notify gulp-autoprefixer
18. В файле gulpfile.js - включаем строгий режим - вначале файла помещаем строку - 'user strict'
и пишем наши задания - task:

```
var gulp = require ('gulp');
var gp = require ('gulp-load-plugins')();

gulp.task('sass', function () {
   return gulp.src('./source/style/app.scss')
      .pipe(gp.sourcemaps.init())
      .pipe(gp.sass())
      .on('error',gp.notify.onError({
         title: 'Style'
      }))
      .pipe(gp.autoprefixer({
         browsers: [
            'last 3 varsion',
            '> 1 %',
            'ie 8',
            'ie 9',
            'Opera 12.1'
         ]
   }))
   .pipe(gp.sourcemaps.write())
   .pipe(gulp.dest('./build/assets/css'))
});
```

19. Запускаем процесс выполнения заданий - gulp sass

Формируем МОДУЛИ - tesks - заданий
20. Создадим папку где будем формировать модули заданий - mkdir -p gulp/tasks

21. Первый модуль sass.js - touch gulp/tasks/sass.js
В этот модуль-файл переносим код задания - task - sass,
оборачиваем его в самовызывющюся функцию и присваиваем его modul.export:

```
modul.exports = function () {
   gulp.task('sass', function () {
   return gulp.src('./source/style/app.scss')
      .pipe(gp.sourcemaps.init())
      .pipe(gp.sass())
      .on('error',gp.notify.onError({
         title: 'Style'
```

```
      }))
      .pipe(gp.autoprefixer({
         browsers: [
            'last 3 version',
            '> 1%',
            'ie 8',
            'ie 9',
            'Opera 12.1'
         ]
   }))
   .pipe(gp.sourcemaps.write())
   .pipe(gulp.dest('./build/assets/css'))
});

}
```

22. Файл gulpfile.js - переписываем - всем объявляем глобальную переменную - $, которой придаем
свойства вызова пакетов:

```
global.$ = {
    gulp:require ('gulp'),
    gp: require ('gulp-load-plugins')()
}
```

23. И затем в модуле - задаче добавляем ее ко всем операторам вызывающим эти пакеты

```
module.exports = function() {
  $.gulp.task('sass', function() {
    return $.gulp.src('./source/style/app.scss')
     .pipe($.gp.sourcemaps.init())
     .pipe($.gp.sass()).on('error', $.gp.notify.onError({ title: 'Style' }))
     .pipe($.gp.autoprefixer({ browsers: $.config.autoprefixerConfig }))
     .pipe($.gp.sourcemaps.write())
     .pipe($.gulp.dest($.config.root + '/assets/css'))
     .pipe($.browserSync.stream());
  })
};
```

24. Возвращаемся в модуль -файл - gulpfile.js - и вызываем (реквайрим) модель с заданием - sass

```
// реквайрим (вызываем) модуль sass.js
// и сразу делаем вызов (регистрируем -task), чтобы иметь к нему доступ

require('./gulp/tasks/sass.js')();

$.gulp.task('sass');
```

25. Проверяем работу модуля - gulp sass

Создаем еще один модуль задачу - конвертации кода - pug в html
26. Создаем папку - mkdir source/template
27. и в нем создаем файл - index.pug - touch source/template/index.pug
28. устанавливаем пакет - gulp-pug - npm i -D gulp-pug (yarn add gulp-pug)
28. Создаем файл - touch gulp/tasks/pug.js
29. В фай - pug.js - пишем код задания:

```
'use strict' /* строгость скрипта */

module.exports = function () {
   $.gulp.task('pug', function () {
      return $.gulp.src('./source/template/*.pug')
         .pipe($.gp.pug({pretty: true})) // true - чтобы HTML
                                        // компилировался не
                                       //  в одну строку
         .on('error', $.gp.notify.onError(function
            (error) {
            return {
               title: 'Pug',
               message: error.message
            }
         }))
         .pipe($.gulp.dest('./build/'));
   });
};
```

30. В файле - index.pug - пишем код аналогичный - html

```
block variables

doctype html
html
  head
    meta(charset="utf-8")
    title #{title}
    meta(content="" name="author")
    meta(content="" name="description")
    meta(content="" name="keywords")
    meta(content="width=device-width, initial-scale=1" name="viewport")
    meta(content="ie=edge" http-equiv="x-ua-compatible")

    //- favicon block start
    //- favicon block end

    link(rel='stylesheet' href='/assets/css/foundation.css')
    link(rel='stylesheet' href='/assets/css/app.css')
    script(src='/assets/js/foundation.js' defer)
    script(src='/assets/js/app.js' defer)
```

```
  <!--[if lt IE 9]>
  script(src="http://cdnjs.cloudflare.com/ajax/libs/html5shiv/3.7.2/html5shiv.min.js")
  <![endif]-->

 body
      block content

       h1 my page
```

31. Возвращаемся в файл - gulpfile.js - и реквайрим и вызываем модуль

```
require('./gulp/tasks/pug.js')();
$.gulp.task('pug');
```

Убираем повторяющиеся пути :

32. Создаем папку - puth - mkdir gulp/puths
33. В ней создаем файл - tasks.js - touch gulp/puths/tasks.js
этом файле мы будем собирать вызов всех заданий

```
'use strict' /* строгость скрипта */

module.exports = [
   './gulp/tasks/sass.js',
   './gulp/tasks/pug.js',
];
```

34. А файл - gulpfile.js - перепишем - добавим в глобальные переменные - методы - path и task
и через них зададим функцию - forEach - которая будет пробегать массив из заданий, записанных в
модуле - tasks.js и будет - реквайрить каждое из заданий

```
global.$ = {
    path: {
        task: require('./gulp/paths/tasks.js')
    },
    gulp:require ('gulp'),
    gp: require ('gulp-load-plugins')()
}

// реквайрим (вызываем) модуль sass.js
// и сразу делаем вызов (регистрируем -task), чтобы иметь к нему доступ

$.path.task.forEach(function (taskPath) {
    require(taskPath)();
});
```

Автоматизируем процесс удаления - папки - build

35. Инсталируем плагин - del - npm install -D del (yarn add del)
36. Создаем файл - touch gulp/tasks/clean.js и пишем в нее код:

```
'use strict' /* строгость скрипта */

module.exports = function () {
   $.gulp.task('clean', function () {
      return $.del([
         'build'
      ]);
   });
};
```

36. Записываем это задание в модуль - tasks.js - './gulp/tasks/clean.js',
37. В в gulpfile.js - впишем вызов - реквайер пакета - del в глобальную переменную, чтобы вызывать из любого места - del:require ('del'),
39. Запускаем - gulp clean (!!!!)

Устанавливаем BrowserSync

40. Инсталируем его npm i -D browser-synk (yarn add browser-sync)
41. Создаем новый файл - touch gulp/tasks/serve.js и записываем всего код:

```
'use strict' /* строгость скрипта */

module.exports = function () {
   $.gulp.task('serve', function () {
      s.browserSync.init({
         open: false, // не надо автоматически открывать браузер
         server: './build' // только эту папку будет обслуживать
      })
      $.browserSync.watch('build', $.browserSync.reload);
   }); // BrowserSync - следит только за папкой - build и когда
      // в ней что-то изменяется только тогда он перезагружается
};
```

Прописываем - watcher - он будет следить за изменениями в папке source

42. watch - уже встроен а Gulp - поэтому его не надо специально инсталировать
43. Создаем файл - touch gulp/tasks/watch.js и пишем код:

```
'use strict' /* строгость скрипта */

module.exports = function() {
  $.gulp.task('watch', function() {
   $.gulp.watch('./source/js/**/*.js', $.gulp.series('js:process'));
   $.gulp.watch('./source/style/**/*.scss', $.gulp.series('sass'));
   $.gulp.watch('./source/template/**/*.pug', $.gulp.series('pug'));
   $.gulp.watch('./source/images/**/*.*', $.gulp.series('copy:image'));
  });
};
```

44. дописываем - gulpfile.js:

```
$.gulp.task('default', $.gulp.series(
  'clean',
  $.gulp.parallel(
    'sass',
    'pug'
  ),
  $.gulp.parallel(
    'watch',
    'serve'
  )
));
```

45. Регистрируем задания в модуляциями tasks.js:

```
'./gulp/tasks/serve.js',
    './gulp/tasks/watch.js'
```

46. Добавить в gulpfile.js - глобальную переменную - browserSync: require('browser-sync').create(),

Формирование СПРАЙТА из - png

47. Устанавливаем пакет spritesmith (ковать-собрать вручную -фею-эльфа)
npm i -D gulp.spritesmith (yarn add gulp.spritesmithl)
48. Создаем файл - touch gulp/tasks/sprite.png.js
49. Добавим в файл gulpfile.js - глобальную переменную - spritesmith:require('gulp.spritesmith'),
50. Создадим папку где будут лежать - png - файлы для спрайта - mkdir source/sprite
51. В файле - sprite.png.js - пропишем код:
''use strict' /* строгость скрипта */

```
module.exports = function () {
    $.gulp.task('sprite:png', function () {
      return $.gulp.src('./source/sprite/*.png')// папка откуда забираем
      .pipe($.gp.spritesmith({
        imgName: 'sprite.png',
        cssName: 'sprite.css'
      }))
    .pipe($.gulp.dest('./build/assets/img'));
    });
};
```

52. Добавляем в файл задач- tasks - нашу задачу - './gulp/tasks/sprite.png.js'

Настройка оптимизации - конкатенации файлов - JawaScript -
53. Инсталируем пакет - gulp-concat - npm install gulp-concat —save-dev (yarn add -gulp-concat)

54. Создаем - файл - touch gulp/tasks/js.foundation.js и пишем код:

```
''use strict';

module.exports = function() {
  $.gulp.task('js:foundation', function() {
    return $.gulp.src($.path.jsFoundation) //последовательность объединяемых файлов
      .pipe($.gp.concat('foundation.js')) // название файла в который объединяются
      .pipe($.gulp.dest($.config.root + '/assets/js')) // куда поместить этот файл
  })
};
```

54. Создадим в файл - touch gulp/paths/js.foundation.js и пишем код:

```
'use strict';

module.exports = [
  './node_modules/jquery/dist/jquery.min.js'
];
```

55. Импортируем JQUERY - yarn add jquery
56. в файле - gulpfely.js - прописываем в глобальныq переменныq - pyth -

```
jsFoundation: require('./gulp/paths/js.foundation.js'),
```

57. В файле - tasks.js - прописываем эту задачу - .’/gulp/tasks/js.foundation.js',
58. Создаем новую папку в mkdir source/js
59. В ней создаем файл - touch source/js/app.js и пишем в ней код:
60. В файле gulpfile.js - записываем нашу задачу в параллельный поток - 'js:foundation',
61. Проверяем работу - gulp - работает.

Дорабатываем пути:

62. Создаем файл - confige - touch gulp/config.js и пишем код:

```
module.exports = {
  root: './build',

  autoprefixerConfig: ['last 3 version', '> 1%', 'ie 8', 'ie 9', 'Opera 12.1']
};
```

63. Дописываем в файл - gulpfile.js глобальны переменные

```
package: require('./package.json'),
  config: require('./gulp/config'),
```

64. переписываем - модуль - serve.js

```
module.exports = function() {
  $.gulp.task('serve', function() {
    $.browserSync.init({
      open: false,
      server: $.config.root
    });

    $.browserSync.watch([$.config.root + '/**/*.*', '!**/*.css'], $.browserSync.reload);
  });
```

```
};
```

66. Переписываем модуль - clean.js

```
module.exports = function() {
  $.gulp.task('clean', function(cb) {
    return $.del($.config.root, cb);
  });
};
```

67. В файле tasks/js.foundation.js - исправляем пути

```
 .pipe($.gulp.dest($.config.root + '/assets/js'))
```

68. В файле pug.js - переписываем путь - .pipe($.gulp.dest($.config.root));

69. Переписываем файл - sass.js -

```
'use strict' /* строгость скрипта */

module.exports = function () {
    $.gulp.task('sass', function () {
    return $.gulp.src('./source/style/app.scss')
        .pipe($.gp.sourcemaps.init())
        .pipe($.gp.sass()).on('error',$.gp.notify.onError({title: 'Style'}))
        .pipe($.gp.autoprefixer({ browsers: $.config.autoprefixerConfig }))
        .pipe($.gp.sourcemaps.write())
        .pipe($.gulp.dest($.config.root + '/assets/css'))
        .pipe($.browserSync.stream());
});
}
```

70. Переписываем модуль - sprite.js - меняем путь

```
.pipe($.gulp.dest($.config.root + '/assets/img'))
```

71. переписываем модуль - watch.js - открываем в нем строку с отслеживанием изменений в файлах -js

```
'use strict' /* строгость скрипта */

module.exports = function() {
  $.gulp.task('watch', function() {
    $.gulp.watch('./source/js/**/*.js', $.gulp.series('js:process'));
    $.gulp.watch('./source/style/**/*.scss', $.gulp.series('sass'));
    $.gulp.watch('./source/template/**/*.pug', $.gulp.series('pug'));
   // $.gulp.watch('./source/images/**/*.*', $.gulp.series('copy:image'));
  });
};
```

НЕТ ЕЩЕ РАНО - не дописали.

ПРОПИСЫВАЕМ Конкатенацию - CSS - файлов

72. Загружаем плагин - gulp-concat-css - npm install --save-dev gulp-concat-css

(yarn add gulp-concat-css)
73. Создаем файл touch gulp/tasks/css.foundation.js и пишем в него код:

```
'use strict';

module.exports = function() {
  $.gulp.task('css:foundation', function() {
    return $.gulp.src($.path.cssFoundation)
      .pipe($.gp.concatCss('foundation.css'))
      .pipe($.gp.csso())
      .pipe($.gulp.dest($.config.root + '/assets/css'))
  })
};
```

74. В файле gulpfile.js - добавляем в глобальные переменные - path еще одну переменную cssFoundation: require('./gulp/paths/css.foundation.js'),

75. Создаем файл touch gulp/paths/css.foundation.js и пишем код:

```
'use strict';

module.exports = [
  './node_modules/normalize.css/normalize.css'
];
```

76. Добавляем задачу в tasks.js - './gulp/tasks/css.foundation.js',
77. В файл - gulpfile.js - нашу задачу в параллельную. ветку - 'css:foundation',
78. Устанавливаем normalize.css - npm install --save normalize.css (yarn add normalize.css)
79. Проверяем - gulp css:foundation - работает

Дорабатываем для JS process

80. Создаем файл - touch gulp/tasks/js.process.js и пишем в него код:

```
'use strict';

module.exports = function() {
  $.gulp.task('js:process', function() {
    return $.gulp.src($.path.app)
      .pipe($.gp.sourcemaps.init())
      .pipe($.gp.concat('app.js'))
      .pipe($.gp.sourcemaps.write())
      .pipe($.gulp.dest($.config.root + '/assets/js'))
  })
};
```

81. В файле - gulpfile.js - в глобальную переменную - path записываем новую переменную -
app: require('./gulp/paths/app.js')

82. Создаем файл - touch gulp/paths/app.js и пишем код: - прописываем путь

```
'use strict';

module.exports = [
  './source/js/app.js'
];
```
83. Прописываем задачу в tasks.js - './gulp/tasks/js.process.js',
84.В файл gulpfile.js - прописываем нашу задачу в параллельный процесс
'js:process',
85. Проверяем работу - gulp js:process - работает

Устанавливаем проверку JS - плагин ESLint
86. Инсталирем ESLint плагин - npm install gulp-eslint (yarn add gulp-eslint)
87. Создаем файл - touch gulp/tasks/js.lint.js и пишем код:

```
'use strict';

module.exports = function() {
  $.gulp.task('js:lint', function() {
    return $.gulp.src($.path.app)
      .pipe($.gp.eslint())
      .pipe($.gp.eslint.format());
  })
};
```
88. Прописываем нашу задачу в файле - tasks.js - './gulp/tasks/js.lint.js',
89. Создаем файл в корне проекта - touch .eslintrc - скрытый файл - и пишем в него код:

```
 {
    "env": {
       "browser": true
    },
    "extends": "eslint:recommended",
    "rules": {
       "indent": [2, 2],
       "quotes": [2, "single"],
       "semi": [2, "always"]
    }
}
```
- это правила .
90. Проверяем - gulp js:lint - работает

Создаем задачу для Копирования IMG - файлов
91. Создаем файл - touch gulp/tasks/copy.image.js и пишем в него код:

```
'use strict';

module.exports = function() {
  $.gulp.task('copy:image', function() {
    return $.gulp.src('./source/images/**/*.*', { since: $.gulp.lastRun('copy:image') })
      .pipe($.gulp.dest($.config.root + '/assets/img'));
  });
};
```

92. Прописываем задачу в файл - tasks.js - './gulp/tasks/copy.image.js',
93. В файле - gulpfile.js - в параллельной ветке прописываем нашу задачу: 'copy:image',
94. Создаем папку - mkdir source/images и записываем в него картинки
95.Проверяем gulp copy:image - работает, gulp - работает

Создаем задачу - Генерация Спрайта из SVG - файлов
96. Инсталируем плагин -npm install --save-dev gulp-svg-sprite (yarn add gulp-svg-sprite)
97. Записываем файл - touch gulp/tasks/sprite.svg.js и пишем код:
98. Устанавливаем плагины - gulp-svgmin - npm install gulp-svgmin (yarn add gulp-svgmin) - минимизация
svg - файлов
99. Устанавливаем плагин - gulp-cheerio - удаляет лишние атрибуты из svg - yarn add gulp-cheerio
100. Устанавливаем плагин - gulp-replace — фиксинг некоторых багов yarn add gulp-replace

101. Прописываем задачу в файле tasks.js - './gulp/tasks/sprite.svg.js',
102. Прописываем задачу в gulpfile.js - 'sprite:svg',
103. Создаем папку - mkdir source/sprite и копируем в нее svg - файла
104. Проверяем - gulp sprite:svg

Создаем задачу копирования шрифтов из исходников в продакшен

105. Создаем файл - touch gulp/tasks/copy.font.js - шрифты не надо оптимизировать они уже оптимизированы и поэтому пишем следующий код:

```
'use strict';

module.exports = function() {
  $.gulp.task('copy:font', function() {
    return $.gulp.src('./source/fonts/**/*.*')
      .pipe($.gulp.dest($.config.root + '/assets/fonts'));
  });
};
```

106. Прописываем нашу задачу в файле - tasks.js - './gulp/tasks/copy.font.js'
107. Прописываем задачу в файле -gulp.file.js - 'copy:font'
108. Создаем папку - mkdir source/fonts - и копируем в нее шрифты
109. Проверяем работу - gulp copy:font - работает, gulp - работает.

Создадим задачу, которая копирует все папки файлы из папки - SOURCE, кроме папок - fonts, images, js, sprite, style, template

110. Создаем файл - touch gulp/tasks/copy.other.js и пишем код:

```
'use strict';

module.exports = function() {
  $.gulp.task('copy:other', function() {
    return $.gulp.src(['./source/**/*.*', '!./source/fonts/*.*',
'!./source/images/*.*','!./source/js/*.*',
                  '!./source/sprite/*.*','!./source/style/*.*','!./source/template/*.*'])
```

```
    .pipe($.gulp.dest($.config.root));
        //.pipe($.gulp.dest('./build/'));  });
};
```

111. Прописываем задачу в файле - tasks.js - './gulp/tasks/copy.other.js'
112. Записываем эту же задачу в параллельный поток - 'copy:other'
113. Создаем в папке source - любые паки и файлы и проверяем их копирование в папку продакшн - build - запускаем выполнение нашей задачи - gulp copy:other - проверка показала,
что код надо изменить, чтобы в исключаемых папках не котировались и вложенные папки
114. Измененный код файла - copy.other.js ;

```
'use strict';

module.exports = function() {
  $.gulp.task('copy:other', function() {
    return $.gulp.src(['./source/**/*.*', '!./source/fonts/**/*.*',
'!./source/images/**/*.*','!./source/js/**/*.*',
                  '!./source/sprite/**/*.*','!./source/style/**/*.*','!./source/template/**/*.*'])
     .pipe($.gulp.dest($.config.root));
  });
};
```

115. Проверяем повторно - gulp copy:other - работает, проверяем - gulp - проверка показала, что
задача - pug - не все файлы конвертирует в html - и не все файлы копирует в папку - build - пришлось
исправить код в файле - pug.js - вот он исправленный:

```
'use strict' /* строгость скрипта */

module.exports = function () {
   $.gulp.task('pug', function () {
      return $.gulp.src('./source/template/**/*.pug')
         .pipe($.gp.pug({pretty: true})) // true - чтобы HTML
                                        // компилировался не
                                        //  в одну строку
         .on('error', $.gp.notify.onError(function
            (error) {
            return {
               title: 'Pug',
               message: error.message
            }

         }))
         .pipe($.gulp.dest($.config.root));
         //.pipe($.gulp.dest('./build/'));
   });
};
```

116. Дорабатываем Задачу - watch - отслеживающую изменения вносимые в результате разработки пишем код:

```
'use strict' /* строгость скрипта */

module.exports = function() {
  $.gulp.task('watch', function() {
    $.gulp.watch('./source/js/**/*.js', $.gulp.series('js:process'));
    $.gulp.watch('./source/style/**/*.scss', $.gulp.series('sass'));
    $.gulp.watch('./source/template/**/*.pug', $.gulp.series('pug'));
    $.gulp.watch('./source/images/**/*.*', $.gulp.series('copy:image'));
    $.gulp.watch('./source/fonts/**/*.*', $.gulp.series('copy:font'));
    $.gulp.watch('./source/sprite/**/*.*', [$.gulp.series('sprite:svg'),
                                    $.gulp.series('sprite:png')]);

  });
};
```

117. Проверяем - gulp - проверка показала, что надо исправить код - не правильно запуск задач - sprite:svg и sprite:png - исправляем:

```
'use strict' /* строгость скрипта */

module.exports = function() {
  $.gulp.task('watch', function() {
    $.gulp.watch('./source/js/**/*.js', $.gulp.series('js:process'));
    $.gulp.watch('./source/style/**/*.scss', $.gulp.series('sass'));
    $.gulp.watch('./source/template/**/*.pug', $.gulp.series('pug'));
    $.gulp.watch('./source/images/**/*.*', $.gulp.series('copy:image'));
    $.gulp.watch('./source/fonts/**/*.*', $.gulp.series('copy:font'));
    $.gulp.watch('./source/sprite/**/*.*', $.gulp.series(['sprite:svg','sprite:png']));

  });
};
```

118. Проверяем - gulp - работает - добаяляем файла в папку - sprite - изменения отслеживаются и
спрайты из svg и из png - формируются.

119. Теперь в файл watch - добавляем отслеживание изменений во всех папках source и запускаем задачу copy;other - проводит изменения во всех папках, кроме оговоренных:

```
'use strict' /* строгость скрипта */

module.exports = function() {
  $.gulp.task('watch', function() {
    $.gulp.watch('./source/js/**/*.js', $.gulp.series('js:process'));
    $.gulp.watch('./source/style/**/*.scss', $.gulp.series('sass'));
    $.gulp.watch('./source/template/**/*.pug', $.gulp.series('pug'));
    $.gulp.watch('./source/images/**/*.*', $.gulp.series('copy:image'));
    $.gulp.watch('./source/fonts/**/*.*', $.gulp.series('copy:font'));
    $.gulp.watch('./source/sprite/**/*.*', $.gulp.series(['sprite:svg','sprite:png']));
    $.gulp.watch('./source/**/*.*', $.gulp.series('copy:other'));
```

```
  });
};
```

Да, конечно, если мы сделаем изменения в запрещенных папках для задачи - copy:other - задача
copy:other все равно запуститься, но не будет вносить изменения в эти запрещенные файлы и папки,
для внесения этих изменений запустятся соответствующие задачи.

ПОКА ВСЕ!!! Считаю, что СБОРКА собрана. Если в ходе реализации проектов возникнет необходимость, буду вносить изменения в эту СБОРКУ.

Отправляем СБОРКУ в Репозитарий

120. git status
121. git add .
122. git commit -m ‘my first sborka’
123. git push