第五节 Grunt

1.介绍

- Grunt介绍
  - 中文主页：http://www.gruntjs.net/
  - 是一套前端自动化构建工具，一个基于nodeJs的命令行工具
  - 它是一个任务运行器, 配合其丰富强大的插件
  - 常用功能:
    - 合并文件(js/css)
    - 压缩文件(js/css)

        - 语法检查(js)
        - less/sass预编译处理
        - 其它...
- Grunt插件介绍
    - grunt官网的插件列表页面 http://www.gruntjs.net/plugins
    - 插件分类:
        - grunt团队贡献的插件：插件名大都以contrib-开头
        - 第三方提供的插件：大都不以contrib-开头
    - 常用的插件:
        - grunt-contrib-clean——清除文件(打包处理生成的)
        - grunt-contrib-concat——合并多个文件的代码到一个文件中
        - grunt-contrib-uglify——压缩js文件
        - grunt-contrib-jshint——javascript语法错误检查；
        - grunt-contrib-cssmin——压缩/合并css文件
        - grunt-contrib-htmlmin——压缩html文件
        - grunt-contrib-imagemin——压缩图片文件(无损)
        - grunt-contrib-copy——复制文件、文件夹
        - grunt-contrib-watch——实时监控文件变化、调用相应的任务重新执行

2.简单使用

2.1安装grunt,

```
yarn add grunt -D
```

2.2 创建一个简单的应用grunt_test

```
|- build----------构建生成的文件所在的文件夹
|- src------------源码文件夹
|- js---------------js源文件夹
|- css--------------css源文件夹
|- index.html-----页面文件
|- Gruntfile.js---grunt配置文件(注意首字母大写)
|- package.json---项目包配置文件
  {
  "name": "grunt_test",
  "version": "1.0.0"
  }
```

2.3 运行构建项目命令及配置文件

- 命令: grunt //提示成功, 但没有任何效果(还没有使用插件定义任务)
- 也可以使用grunt 任务名称

2.4 配置文件: gruntfile.js

- 此配置文件本质就是一个node函数类型模块
- 配置编码包含3步:
  i. 初始化插件配置
  ii. 加载插件任务
  iii. 注册构建任务
- 基本编码:

```
// Grunt 的入口文件
// 用于定义一些需要Grunt自动执行的任务
// 需要导出一个函数
// 此函数接收一个grunt的形参,内部提供一些创建任务是可以用到的api

module.exports = grunt => {
    grunt.registerTask('foo',() => {
        console.log("hello grunt");
    })
}
```

2.4.1 registerTask

- 参数1是任务的名称
- 参数2是如果是字符串这是任务描述,如果是函数就是需要执行的任务

```
    // 任务描述(可以通过grunt --help查看任务描述)
    grunt.registerTask('bar','描述信息', () => {
        console.log('other task~');
    })
    // 默认任务
    grunt.registerTask('default',['foo','bar'])
```

- 如果任务名称是'default',则只需运行npx grunt就可以执行default任务

- 如果参数2是一个数据,那就是依次执行数组中所有任务
- 处理异步任务,需要async()函数

```
    // 执行一步任务
        // 不会生效
    grunt.registerTask('async-task',() => {
        setTimeout(() => {
            console.log('async task, working');
        },1000)
    })
    grunt.registerTask('default',function(){
        // 需要生成一个回调函数
        const done = this.async()
        setTimeout(() => {
            console.log('async task, working');
          // 执行完成之后会执行该函数,标明任务完成
            done()
        }, 1000);
    })
```

2.4.2 标记任务失败

```
module.exports = grunt => {
  // 任务函数执行过程中如果返回 false
  // 则意味着此任务执行失败
  grunt.registerTask('bad', () => {
    console.log('bad working~')
    return false
  })

  grunt.registerTask('foo', () => {
    console.log('foo working~')
  })

  grunt.registerTask('bar', () => {
    console.log('bar working~')
  })

```

```
  // 如果一个任务列表中的某个任务执行失败
  // 则后续任务默认不会运行
  // 除非 grunt 运行时指定 --force 参数强制执行
  grunt.registerTask('default', ['foo', 'bad', 'bar'])

  // 异步函数中标记当前任务执行失败的方式是为回调函数指定一个 false 的实参
  grunt.registerTask('bad-async', function () {
    const done = this.async()
    setTimeout(() => {
      console.log('async task working~')
      done(false)
    }, 1000)
  })
}
```

2.4.3 初始化插件配置

```
// 1. 初始化插件配置
grunt.initConfig({
    //主要编码处
});
```

- 单任务的参数

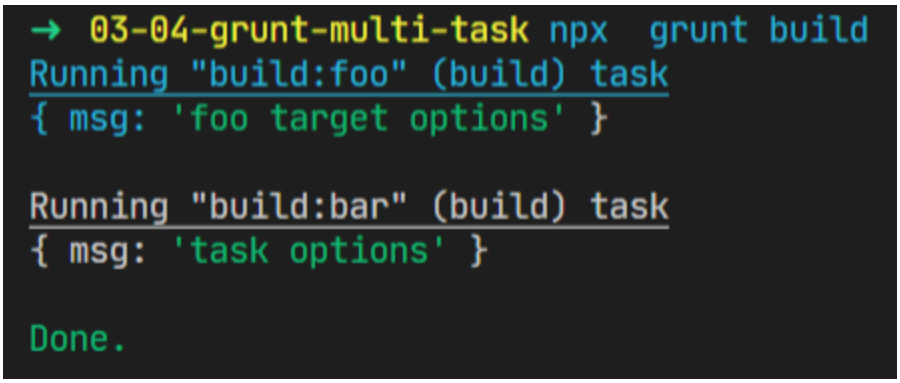
```
module.exports = grunt => {
  // 单任务形式,配置参数
  grunt.initConfig({
    foo: {
      bar: 123
    }
  })
  // 使用grunt.config(key) 获取指定参数
  grunt.registerTask('foo', () => {
    console.log(grunt.config('foo.bar'));
  })
}
```

- 多任务的参数
- 运行所有任务yarn grunt build
- 运行一个任务yarn grunt build:foo
- 在build中定义的属性都会成为一个子任务,除了option以外

```
module.exports = grunt => {
  // 多目标模式，可以让任务根据配置形成多个子任务registerMultiTask


  grunt.initConfig({
    build: {
      options: {
        msg: 'task options'
      },
      foo: {
        options: {
          msg: 'foo target options'
        }
      },
      bar: '456'
    }
  })

  grunt.registerMultiTask('build', function () {
    console.log(this.options())
  })
}
```



```
grunt.initConfig({
  build: {
    css: '1',
    js: '2'
  }
})
grunt.registerMultiTask('build',function() {
  console.log(`target : ${this.target},data: ${this.data}`);
})
```

执行npx grunt build

- this.target返回执行的子任务名称
- this.data返回配置数据

```
→ 03-04-grunt-multi-task npx grunt build
Running "build:css" (build) task
target : css,data: 1

Running "build:js" (build) task
target : js,data: 2
```

2.4.4 加载插件任务

```
// 2. 加载插件任务
// grunt.loadNpmTasks('grunt-contrib-concat');
```

3.常用插件

3.1 合并js: 使用concat插件

- 命令:

  npm install grunt-contrib-concat --save-dev
- 编码:

```
//src/js/test1.js
  (function () {
  function add(num1, num2) {
  return num1 + num2;
```

```
  }
  console.log(add(10, 20));
})();

//src/js/test2.js
  (function () {
  var arr = [2,3,4].map(function (item, index) {
  return item+1;
  });
  console.log(arr);
})();
```

- 配置: Gruntfile.js
  - 配置任务:

```
concat: {
  options: { //可选项配置
    separator: ';'   //使用;连接合并
    },
    build: { //此名称任意
    src:  ["src/js/*.js"],  //合并哪些js文件
    dest: "build/js/built.js" //输出的js文件
  }
}
```

  - 加载插件:
    grunt.loadNpmTasks('grunt-contrib-concat');
  - 注册任务:
    grunt.registerTask('default', ['concat']);
  - 命令:
    grunt //会在build下生成一个built.js

3.2 压缩js: 使用uglify插件

- 下载
  npm install grunt-contrib-uglify --save-dev
- 配置: Gruntfile.js
  - 配置任务:

```
    pkg : grunt.file.readJSON('package.json'),
    uglify : {
        options: {  //不是必须的
        banner: '/*! <%= pkg.name %> - v<%= pkg.version %> - ' +
        '<%= grunt.template.today("yyyy-mm-dd") %> */'
    },
    build: {
        files: {
        'build/js/built-<%=pkg.name%>-<%=pkg.version%>.min.js': ['build/js/built.js']
        }
    }
    }
```

- 加载任务:
  grunt.loadNpmTasks('grunt-contrib-uglify');
- 注册任务:
  grunt.registerTask('default', ['concat', 'uglify']);
- 命令:
  grunt //会在build下生成一个压缩的js文件

3.3 js语法检查: 使用jshint插件

- 命令:
  npm install grunt-contrib-jshint --save-dev
- 编码: .jshintrc

```
{
  "curly": true,
  "eqeqeq": true,
  "eqnull": true,
  "expr" : true,
  "immed": true,
  "newcap": true,
  "noempty": true,
  "noarg": true,
  "regexp": true,
  "browser": true,
  "devel": true,
  "node": true,
```

```
  "boss": false,
}
```

```
  //不能使用未定义的变量
  "undef": true,
  //语句后面必须有分号
  "asi": false,
  //预定义不检查的全局变量
  "predef": [ "define", "BMap", "angular", "BMAP_STATUS_SUCCESS"]
}
```

- 修改src/js/test1.js

```
(function () {
  function add(num1, num2) {
    num1 = num1 + num3
    return num1 + num2;
  }
  console.log(add(10, 20));
})();
```

- 配置：Gruntfile.js
  - 配置任务:

```
jshint : {
    options: {
    jshintrc : '.jshintrc' //指定配置文件
  },
  build : ['Gruntfile.js', 'src/js/*.js'] //指定检查的文件
}
```

  - 加载任务:
    grunt.loadNpmTasks('grunt-contrib-jshint');
  - 注册任务:
    grunt.registerTask('default', ['concat', 'uglify', 'jshint']);
  - 命令:
    grunt //提示变量未定义和语句后未加分号 -->修改后重新编译

3.4 使用cssmin插件

- 安装:
  npm install grunt-contrib-cssmin --save-dev
- 编码:

```
/* test1.css */
#box1 {
  width: 100px;
  height: 100px;
  background: red;
}
/* test2.css */
#box2 {
  width: 200px;
  height: 200px;
  background: blue;
}
```

- 配置：Gruntfile.js
  - 配置任务:

```
cssmin:{
  options: {
    shorthandCompacting: false,
    roundingPrecision: -1
  },
  build: {
    files: {
        'build/css/output.min.css': ['src/css/*.css']
    }
  }
}
```

  - 加载任务:
    grunt.loadNpmTasks('grunt-contrib-cssmin');

- 注册任务:
  grunt.registerTask('default', ['concat', 'uglify', 'jshint', 'cssmin']);
- 命令:
  grunt //在build/css/下生成output.min.css

3.5 使用watch插件（真正实现自动化）

- 命令: npm install grunt-contrib-watch --save-dev
- 配置：Gruntfile.js
  - 配置任务:

```
        watch: {
            scripts: {
                files: ['**/*.js'], // 监视的文件
                tasks: ['babel'], // 文件改变之后要执行的任务呢
                options: {
                    spawn: false, // 是否派进程中运行的任务
                },
            },
            sass: {
                files: ['**/*.js'],
                tasks: ['sass']
            }
        },
```

  - 加载任务:

```
grunt.loadNpmTasks('grunt-contrib-watch');
```

  - 注册任务:

```
grunt.registerTask('default', ['concat', 'uglify', 'jshint', 'watch']
);
改进: grunt.registerTask('myWatch', ['default','watch']);
```

- 命令: grunt //控制台提示watch已经开始监听, 修改保存后自动编译处理

3.6 清除临时文件

安装

```
yarn add grunt-contrib-clean
```

使用

```
module.exports = grunt => {
    grunt.initConfig({
        // 配置子任务,并添加配置选项
        clean: {
            temp: 'temp/app.js' // 可以使用通配符* / **
        }
    })
  // 加载插件
    grunt.loadNpmTasks('grunt-contrib-clean')
}
```

3.7 sass编译

安装

```
yarn add grunt grunt-sass sass node-sass -D
```

使用配置文件

```
const sass = require('node-sass');
module.exports = grunt => {
    grunt.initConfig({
        sass: {
            options: {
                implementation: sass,
                sourceMap: true,
            },
            dist: {
                files: {
                    'dist/css/main.css': 'src/sass/main.scss',
                },
            },
        },
```

```
    });
    grunt.loadNpmTasks('grunt-sass')
}
```

3.8 babel转换js语法

安装插件

```
yarn add grunt-babel @babel/core @babel/preset-env
```

```
const sass = require('node-sass');
module.exports = grunt => {
    grunt.initConfig({
        sass: {
            options: {
                implementation: sass,
                sourceMap: true,
            },
            dist: {
                files: {
                    'dist/css/main.css': 'src/sass/main.scss',
                },
            },
        },
        babel: {
            options: {
                sourceMap: true,
                presets: ['@babel/preset-env'],
            },
            dist: {
                files: {
                    'dist/app.js': 'src/app.js',
                },
            },
        },
    });
    grunt.loadNpmTasks('grunt-sass')
    grunt.loadNpmTasks('grunt-babel')
```

```
    grunt.registerTask('default', ['babel','sass']);
}
```

3.9解决加载模块过多

安装

```
yarn add load-grunt-tasks
```

```
const loadGruntTasks = require('load-grunt-tasks');
const sass = require('node-sass');
module.exports = grunt => {
    grunt.initConfig({
        sass: {
            options: {
                implementation: sass,
                sourceMap: true,
            },
            dist: {
                files: {
                    'dist/css/main.css': 'src/sass/main.scss',
                },
            },
        },
        babel: {
            options: {
                sourceMap: true,
                presets: ['@babel/preset-env'],
            },
            dist: {
                files: {
                    'dist/app.js': 'src/app.js',
                },
            },
        },
    });
    // 自动加载安装的所有的grunt插件任务
    loadGruntTasks(grunt)
```

```
30      // grunt.loadNpmTasks('grunt-sass')
31      // grunt.loadNpmTasks('grunt-babel')
32      grunt.registerTask('default', ['babel','sass']);
33  }
```