8.17. Сборка проекта

На лайве этого раздела мы написали код для сборки проекта. Ниже мы предоставляем исходный код сборки для того, чтобы вы смогли сравнить его со своим кодом. В конце статьи показана сборка для node.js 12 версии.

Сборка на Less

package.json

```
{
  "private": true,
  "devDependencies": {
    "@htmlacademy/editorconfig-cli": "1.0.0",
    "autoprefixer": "10.4.0",
    "browser-sync": "2.27.7",
    "del": "6.0.0",
    "gulp": "4.0.2",
    "gulp-less": "5.0.0",
    "gulp-libsquoosh": "^1.0.12",
    "gulp-plumber": "1.2.1",
    "gulp-postcss": "9.0.1",
    "gulp-rename": "2.0.0",
    "gulp-svgmin": "^4.1.0",
    "gulp-svgstore": "8.0.0",
    "gulp-terser": "2.1.0",
    "postcss": "8.3.11",
    "postcss-csso": "5.0.1",
    "stylelint": "13.13.1",
```

```
    "stylelint-config-htmlacademy": "^0.1.11"
  },
  "scripts": {
    "editorconfig": "editorconfig-cli",
    "stylelint": "stylelint \"source/less/**/*.less\" --syntax less",
    "lint": "npm run editorconfig && npm run stylelint",
    "build": "gulp build",
    "start": "gulp"
  },
  "browserslist": [
    "last 2 versions",
    "not dead",
    "not ie <= 11",
    "not op_mini all",
    "not < 0.25%"
  ],
  "editorconfig-cli": [
    "*.json",
    "*.js",
    "source/*.html",
    "source/js/**/*.js",
    "source/img/**/*.svg",
    "source/less/**/*.less"
  ],
  "engines": {
    "node": "16.13.0"
  },
  "type": "module"
}
```

gulpfile.js

```
import gulp from 'gulp';
import plumber from 'gulp-plumber';
import less from 'gulp-less';
import postcss from 'gulp-postcss';
import autoprefixer from 'autoprefixer';
import csso from 'postcss-csso';
```

```
import rename from 'gulp-rename';
import terser from 'gulp-terser';
import squoosh from 'gulp-libsquoosh';
import svgo from 'gulp-svgmin';
import svgstore from 'gulp-svgstore';
import del from 'del';
import browser from 'browser-sync';

// Styles

export const styles = () => {
return gulp.src('source/less/style.less', { sourcemaps: true })
.pipe(plumber())
.pipe(less())
.pipe(postcss([
autoprefixer(),
csso()
]))
.pipe(rename('style.min.css'))
.pipe(gulp.dest('build/css', { sourcemaps: '.' }))
.pipe(browser.stream());
}

// HTML

const html = () => {
return gulp.src('source/*.html')
.pipe(gulp.dest('build'));
}

// Scripts

const scripts = () => {
return gulp.src('source/js/script.js')
.pipe(gulp.dest('build/js'))
.pipe(browser.stream());
}

// Images

const optimizeImages = () => {
return gulp.src('source/img/**/*.{png,jpg}')
.pipe(squoosh())
.pipe(gulp.dest('build/img'))
}
```

```
const copyImages = () => {
return gulp.src('source/img/**/*.{png,jpg}')
.pipe(gulp.dest('build/img'))
}

// WebP

const createWebp = () => {
return gulp.src('source/img/**/*.{png,jpg}')
.pipe(squoosh({
webp: {}
}))
.pipe(gulp.dest('build/img'))
}

// SVG

const svg = () =>
gulp.src(['source/img/*.svg', '!source/img/icons/*.svg'])
.pipe(svgo())
.pipe(gulp.dest('build/img'));

const sprite = () => {
return gulp.src('source/img/icons/*.svg')
.pipe(svgo())
.pipe(svgstore({
inlineSvg: true
}))
.pipe(rename('sprite.svg'))
.pipe(gulp.dest('build/img'));
}

// Copy

const copy = (done) => {
gulp.src([
'source/fonts/*.{woff2,woff}',
'source/*.ico',
], {
base: 'source'
})
.pipe(gulp.dest('build'))
done();
}
```

```
// Clean

const clean = () => {
return del('build');
};

// Server

const server = (done) => {
browser.init({
server: {
baseDir: 'build'
},
cors: true,
notify: false,
ui: false,
});
done();
}

// Reload

const reload = (done) => {
browser.reload();
done();
}

// Watcher

const watcher = () => {
gulp.watch('source/less/**/*.less', gulp.series(styles));
gulp.watch('source/js/script.js', gulp.series(scripts));
gulp.watch('source/*.html', gulp.series(html, reload));
}

// Build

export const build = gulp.series(
clean,
copy,
optimizeImages,
gulp.parallel(
styles,
html,
scripts,
```

```
svg,
sprite,
createWebp
),
);

// Default
export default gulp.series(
clean,
copy,
copyImages,
gulp.parallel(
styles,
html,
scripts,
svg,
sprite,
createWebp
),
gulp.series(
server,
watcher
));
```

Сборка на SASS

package.json

```
{
  "private": true,
  "devDependencies": {
    "@htmlacademy/editorconfig-cli": "1.0.0",
    "autoprefixer": "10.4.0",
    "browser-sync": "2.27.7",
    "del": "6.0.0",
    "gulp": "4.0.2",
    "gulp-dart-sass": "^1.0.2",
    "gulp-libsquoosh": "^1.0.12",
```

```
    "gulp-plumber": "1.2.1",
    "gulp-postcss": "9.0.1",
    "gulp-rename": "2.0.0",
    "gulp-svgmin": "^4.1.0",
    "gulp-svgstore": "8.0.0",
    "gulp-terser": "2.1.0",
    "postcss": "8.3.11",
    "postcss-csso": "5.0.1",
    "stylelint": "13.13.1",
    "stylelint-config-htmlacademy": "^0.1.11"
  },
  "scripts": {
    "editorconfig": "editorconfig-cli",
    "stylelint": "stylelint \"source/sass/**/*.scss\" --syntax scss",
    "lint": "npm run editorconfig && npm run stylelint",
    "build": "gulp build",
    "start": "gulp"
  },
  "browserslist": [
    "last 2 versions",
    "not dead",
    "not ie <= 11",
    "not op_mini all",
    "not < 0.25%"
  ],
  "editorconfig-cli": [
    "*.json",
    "*.js",
    "source/*.html",
    "source/js/**/*.js",
    "source/img/**/*.svg",
    "source/sass/**/*.scss"
  ],
  "engines": {
    "node": "16.13.0"
  },
  "type": "module"
}
```

gulpfile.js

Node.js 12

Сборка на Less

package.json

```
{
  "private": true,
  "devDependencies": {
    "@htmlacademy/editorconfig-cli": "1.0.0",
    "autoprefixer": "10.2.5",
    "browser-sync": "2.26.14",
    "del": "6.0.0",
    "gulp": "4.0.2",
    "gulp-imagemin": "7.1.0",
    "gulp-less": "4.0.1",
    "gulp-plumber": "1.2.1",
    "gulp-postcss": "9.0.0",
    "gulp-rename": "2.0.0",
    "gulp-sourcemaps": "3.0.0",
    "gulp-svgstore": "7.0.1",
    "gulp-terser": "2.0.1",
    "gulp-webp": "4.0.1",
    "postcss": "8.2.10",
    "postcss-csso": "5.0.1",
    "stylelint": "13.12.0",
    "stylelint-config-htmlacademy": "0.1.4"
  },
  "scripts": {
    "editorconfig": "editorconfig-cli",
    "stylelint": "stylelint \"source/less/**/*.less\" --syntax less",
    "lint": "npm run editorconfig && npm run stylelint",
    "build": "gulp build",
    "start": "gulp"
  },
  "browserslist": [
    "last 2 versions",
    "not dead",
    "not ie <= 11"
  ],
  "editorconfig-cli": [
```

```
        "*.json",
        "*.js",
        "source/*.html",
        "source/js/**/*.js",
        "source/img/**/*.svg",
        "source/less/**/*.less"
      ],
      "engines": {
        "node": "14.15.0"
      }
    }
```

gulpfile.js

```
const gulp = require("gulp");
const plumber = require("gulp-plumber");
const sourcemap = require("gulp-sourcemaps");
const less = require("gulp-less");
const postcss = require("gulp-postcss");
const autoprefixer = require("autoprefixer");
const csso = require("postcss-csso");
const rename = require("gulp-rename");
const terser = require("gulp-terser");
const imagemin = require("gulp-imagemin");
const webp = require("gulp-webp");
const svgstore = require("gulp-svgstore");
const del = require("del");
const sync = require("browser-sync").create();

// Styles

const styles = () => {
return gulp.src("source/less/style.less")
.pipe(plumber())
.pipe(sourcemap.init())
.pipe(less())
.pipe(postcss([
autoprefixer(),
csso()
```

```
]))
.pipe(rename("style.min.css"))
.pipe(sourcemap.write("."))
.pipe(gulp.dest("build/css"))
.pipe(sync.stream());
}

exports.styles = styles;

// HTML

const html = () => {
return gulp.src("source/*.html")
.pipe(gulp.dest("build"));
}

// Scripts

const scripts = () => {
return gulp.src("source/js/script.js")
.pipe(terser())
.pipe(rename("script.min.js"))
.pipe(gulp.dest("build/js"))
.pipe(sync.stream());
}

exports.scripts = scripts;

// Images

const optimizeImages = () => {
return gulp.src("source/img/**/*.{png,jpg,svg}")
.pipe(imagemin([
imagemin.mozjpeg({progressive: true}),
imagemin.optipng({optimizationLevel: 3}),
imagemin.svgo()
]))
.pipe(gulp.dest("build/img"))
}

exports.images = optimizeImages;

const copyImages = () => {
return gulp.src("source/img/**/*.{png,jpg,svg}")
```

```
.pipe(gulp.dest("build/img"))
}

exports.images = copyImages;

// WebP

const createWebp = () => {
return gulp.src("source/img/**/*.{jpg,png}")
.pipe(webp({quality: 90}))
.pipe(gulp.dest("build/img"))
}

exports.createWebp = createWebp;

// Sprite

const sprite = () => {
return gulp.src("source/img/icons/*.svg")
.pipe(svgstore({
inlineSvg: true
}))
.pipe(rename("sprite.svg"))
.pipe(gulp.dest("build/img"));
}

exports.sprite = sprite;

// Copy

const copy = (done) => {
gulp.src([
"source/fonts/*.{woff2,woff}",
"source/*.ico",
"source/img/**/*.svg",
"!source/img/icons/*.svg",
], {
base: "source"
})
.pipe(gulp.dest("build"))
done();
}

exports.copy = copy;
```

```
// Clean

const clean = () => {
return del("build");
};

// Server

const server = (done) => {
sync.init({
server: {
baseDir: "build"
},
cors: true,
notify: false,
ui: false,
});
done();
}

exports.server = server;

// Reload

const reload = (done) => {
sync.reload();
done();
}

// Watcher

const watcher = () => {
gulp.watch("source/less/**/*.less", gulp.series(styles));
gulp.watch("source/js/script.js", gulp.series(scripts));
gulp.watch("source/*.html", gulp.series(html, reload));
}

// Build

const build = gulp.series(
clean,
copy,
optimizeImages,
gulp.parallel(
styles,
```

```
html,
scripts,
sprite,
createWebp
),
);

exports.build = build;

// Default
exports.default = gulp.series(
clean,
copy,
copyImages,
gulp.parallel(
styles,
html,
scripts,
sprite,
createWebp
),
gulp.series(
server,
watcher
));
```

Сборка на SASS

package.json

```
{
  "private": true,
  "devDependencies": {
    "@htmlacademy/editorconfig-cli": "1.0.0",
    "autoprefixer": "10.2.5",
    "browser-sync": "2.26.14",
    "del": "6.0.0",
    "gulp": "4.0.2",
    "gulp-imagemin": "7.1.0",
```

```
    "gulp-sass": "4.1.0",
    "gulp-plumber": "1.2.1",
    "gulp-postcss": "9.0.0",
    "gulp-rename": "2.0.0",
    "gulp-sourcemaps": "3.0.0",
    "gulp-svgstore": "7.0.1",
    "gulp-terser": "2.0.1",
    "gulp-webp": "4.0.1",
    "postcss": "8.2.10",
    "postcss-csso": "5.0.1",
    "stylelint": "13.12.0",
    "stylelint-config-htmlacademy": "0.1.4"
  },
  "scripts": {
    "editorconfig": "editorconfig-cli",
    "stylelint": "stylelint \"source/sass/**/*.scss\" --syntax scss",
    "lint": "npm run editorconfig && npm run stylelint",
    "build": "gulp build",
    "start": "gulp"
  },
  "browserslist": [
    "last 2 versions",
    "not dead",
    "not ie <= 11"
  ],
  "editorconfig-cli": [
    "*.json",
    "*.js",
    "source/*.html",
    "source/js/**/*.js",
    "source/img/**/*.svg",
    "source/sass/**/*.scss"
  ],
  "engines": {
    "node": "14.15.0"
  }
}
```

gulpfile.js

```
const gulp = require("gulp");
const plumber = require("gulp-plumber");
const sourcemap = require("gulp-sourcemaps");
const sass = require("gulp-sass");
const postcss = require("gulp-postcss");
const autoprefixer = require("autoprefixer");
const csso = require("postcss-csso");
const rename = require("gulp-rename");
const terser = require("gulp-terser");
const imagemin = require("gulp-imagemin");
const webp = require("gulp-webp");
const svgstore = require("gulp-svgstore");
const del = require("del");
const sync = require("browser-sync").create();

// Styles

const styles = () => {
return gulp.src("source/sass/style.scss")
.pipe(plumber())
.pipe(sourcemap.init())
.pipe(sass())
.pipe(postcss([
autoprefixer(),
csso()
]))
.pipe(rename("style.min.css"))
.pipe(sourcemap.write("."))
.pipe(gulp.dest("build/css"))
.pipe(sync.stream());
}

exports.styles = styles;

// HTML

const html = () => {
return gulp.src("source/*.html")
.pipe(gulp.dest("build"));
}

// Scripts
```

```
const scripts = () => {
return gulp.src("source/js/script.js")
.pipe(terser())
.pipe(rename("script.min.js"))
.pipe(gulp.dest("build/js"))
.pipe(sync.stream());
}

exports.scripts = scripts;

// Images

const optimizeImages = () => {
return gulp.src("source/img/**/*.{png,jpg,svg}")
.pipe(imagemin([
imagemin.mozjpeg({progressive: true}),
imagemin.optipng({optimizationLevel: 3}),
imagemin.svgo()
]))
.pipe(gulp.dest("build/img"))
}

exports.images = optimizeImages;

const copyImages = () => {
return gulp.src("source/img/**/*.{png,jpg,svg}")
.pipe(gulp.dest("build/img"))
}

exports.images = copyImages;

// WebP

const createWebp = () => {
return gulp.src("source/img/**/*.{jpg,png}")
.pipe(webp({quality: 90}))
.pipe(gulp.dest("build/img"))
}

exports.createWebp = createWebp;

// Sprite

const sprite = () => {
return gulp.src("source/img/icons/*.svg")
```

```
.pipe(svgstore({
inlineSvg: true
}))
.pipe(rename("sprite.svg"))
.pipe(gulp.dest("build/img"));
}

exports.sprite = sprite;

// Copy

const copy = (done) => {
gulp.src([
"source/fonts/*.{woff2,woff}",
"source/*.ico",
"source/img/**/*.svg",
"!source/img/icons/*.svg",
], {
base: "source"
})
.pipe(gulp.dest("build"))
done();
}

exports.copy = copy;

// Clean

const clean = () => {
return del("build");
};

// Server

const server = (done) => {
sync.init({
server: {
baseDir: "build"
},
cors: true,
notify: false,
ui: false,
});
done();
}
```

```
exports.server = server;

// Reload

const reload = (done) => {
sync.reload();
done();
}

// Watcher

const watcher = () => {
gulp.watch("source/sass/**/*.scss", gulp.series(styles));
gulp.watch("source/js/script.js", gulp.series(scripts));
gulp.watch("source/*.html", gulp.series(html, reload));
}

// Build

const build = gulp.series(
clean,
copy,
optimizeImages,
gulp.parallel(
styles,
html,
scripts,
sprite,
createWebp
),
);

exports.build = build;

// Default
exports.default = gulp.series(
clean,
copy,
copyImages,
gulp.parallel(
styles,
html,
scripts,
sprite,
createWebp
```

```
),
gulp.series(
server,
watcher
));
```

Прочитали статью?

Нажмите кнопку «Готово», чтобы сохранить прогресс.

Готово

Если вы обнаружили ошибку или неработающую ссылку, выделите ее и нажмите Ctrl + Enter

Поиск по материалам

Git

Все материалы

В самом начале

- [ ] Пройдите опрос
- [ ] Укажите персональные данные
- [ ] Изучите регламент
- [ ] Прочитайте FAQ
- [ ] Добавьте свой Гитхаб
- [ ] Выберите наставника
- [ ] Создайте проект

Мой наставник

Выбрать наставника

Работа с наставником

У вас осталось **10** из 10 консультаций.

История




Практикум

Тренажёры

Подписка

Для команд и компаний

Учебник по PHP

Профессии

Фронтенд-разработчик

JavaScript-разработчик

Фулстек-разработчик

Курсы

HTML и CSS. Профессиональная вёрстка сайтов

HTML и CSS. Адаптивная вёрстка и автоматизация

JavaScript. Профессиональная разработка веб-интерфейсов

JavaScript. Архитектура клиентских приложений

React. Разработка сложных клиентских приложений

Node.js. Профессиональная разработка REST API

Node.js и Nest.js. Микросервисная архитектура

TypeScript. Теория типов

Алгоритмы и структуры данных

Паттерны проектирования

Webpack

Vue.js 3. Разработка клиентских приложений

Git и GitHub

Анимация для фронтендеров

Блог

С чего начать

Шпаргалки для разработчиков

Отчеты о курсах

Услуги

Работа наставником

Для учителей

Стать автором

Информация

Об Академии

О центре карьеры

Остальное

Написать нам

Мероприятия

Форум

Соглашение

Конфиденциальность

Сведения об образовательной организации

Лицензия № 4696

Sk
Участник