

nicothin / NTH-start-project

Стартовый репозиторий для новых HTML/CSS проектов с SCSS <http://nicothin.github.io/NTH-start-p...>

#starter-kit #scss #css #bem #postcss #gulp #html #starter-project #starter #starterkit

517 commits

6 branches

0 releases

2 contributors

WTFPL

Branch: master

New pull request

Create new file

Upload files

Find file

Clone or download



Рефакторинг глобального JS-файла

Latest commit 5d0b87a 13 days ago

build	start	2 years ago
src	Рефакторинг глобального JS-файла	13 days ago
.editorconfig	add	a year ago
.gitignore	Update .gitignore	5 months ago
.npmrc	Успокоение ошибок npm	a year ago
.stylelintignore	Добавил стайллинг	8 months ago
.stylelintrc	Обновил зависимости	18 days ago
.yaspellerrc	Добавил про object-fit-images полифил в описание	5 months ago
LICENSE	Добавил лицензию	8 months ago
README.md	Update README.md	2 months ago
createBlock.js	Рефакторинг: БОльшая аккуратность	2 months ago
gulpfile.js	Рефакторинг: описание действий в галпфайле	13 days ago
package.json	Обновил зависимости	18 days ago
projectConfig.json	Добавил блок для адаптирующегося медиаконтента	5 months ago

README.md

Проект с gulp

devDependencies up to date

dependencies up to date

Команда	Результат
npm i	Установить зависимости
npm start	Запустить сборку, сервер и слежение за файлами
npm start ЗАДАЧА	Запустить задачу с названием ЗАДАЧА (список задач в gulpfile.js)
npm run build	Сборка проекта без карт кода (сжатый вид, как результат работы)
npm run deploy	Сборка проекта без карт кода и отправка содержимого папки сборки на github-pages
npm run test:style	Проверка стилевой составляющей проекта stylelint
npm run test:orpho	Поиск ошибок и опечаток в .html и .md файлах в ./src с помощью yaspeller

Если при вызове `npm start` ничего не происходит, смотрите `./projectConfig.json`, он содержит синтаксическую ошибку. Проверить можно [линтером](#).

Предполагается, что все команды вы выполняете в `bash` (для OSX и Linux это самый обычный встроенный терминал, для Windows это, к примеру, Git Bash). В Windows установку пакетов (`npm i`) нужно выполнять в терминале, запущенном от имени администратора.

Парадигма

- Используется именование классов, файлов и переменных по БЭМ.
- Список использованных в проекте БЭМ-блоков и доп. файлов указан в `./projectConfig.json`.

- Каждый БЭМ-блок в своей папке внутри `./src/blocks/` (стили, js, картинки, разметка; обязателен только стиливой файл).
- Есть глобальные файлы: стиливые, js, шрифты, картинки.
- Диспетчер подключения стилей `./src/scss/style.scss` генерируется автоматически при старте любой gulp-задачи.
- Для разметки можно использовать [микрошаблонизацию](#). А можно и не использовать.
- Перед созданием коммита запускается проверка стиливых файлов, входящих в коммит.

ЧАВО

- [Как правильно организовать сборку целой страницы](#)
- [Как добавлять плагины jQuery](#)

Есть вопрос — задайте его в [issues](#).

Стили

Файл-диспетчер подключений (`./src/scss/style.scss`) формируется автоматически на основании указанных в `./projectConfig.json` блоков и доп. файлов. Писать в `./src/scss/style.scss` что-либо руками бессмысленно: при старте автоматизации файл будет перезаписан.

Используемый постпроцессинг:

1. [autoprefixer](#)
2. [css-mqpacker](#)
3. [postcss-import](#)
4. [postcss-inline-svg](#)
5. [gulp-cleancss](#) (только в режиме сборки без карт кода)
6. [postcss-object-fit-images](#) (в паре с [полифилом](#))
7. [postcss-image-inliner](#)

Для [postcss-image-inliner](#) указано ограничение на размер файла в 10 Кб, файлы ищутся в `src/blocks/**/*.img_to_bg/`. Чтобы избежать конфликтов имен, добавляйте к именам изображений префикс (имя блока), например: `src/blocks/mega-block/img_to_bg/mega-block__avatar.png`

Блоки

Каждый блок лежит в `./src/blocks/` в своей папке. Каждый блок — как минимум, папка и одноимённый scss-файл.

Возможное содержимое блока:

<code>demo-block/</code>	# Папка блока
<code>img/</code>	# Изображения, используемые блоком и обрабатываемые автоматикой сборки
<code>some-folder/</code>	# Какая-то сторонняя папка, не обрабатываемая автоматикой
<code>demo-block.scss</code>	# Стиливой файл блока
<code>demo-block--mod.scss</code>	# Отдельный стиливой файл БЭМ-модификатора блока
<code>demo-block.js</code>	# js-файл блока
<code>demo-block--mod.js</code>	# js-файл для отдельного БЭМ-модификатора блока
<code>demo-block.html</code>	# Варианты разметки (как документация блока или как вставляемый микрошаблонизатором фрагмент)
<code>readme.md</code>	# Какое-то пояснение

Подключение блоков

Список используемых блоков и дополнительных подключаемых файлов указан в `./projectConfig.json`. Список файлов и папок, взятых в обработку можно увидеть в терминале, если раскомментировать строку `console.log(list);` в `gulpfile.js`.

blocks

Объект с блоками, используемыми на проекте. Каждый блок — отдельная папка с файлами, по умолчанию лежат в `./src/blocks/`.

Каждое подключение блока — массив, который можно оставить пустым или указать файлы элементов или модификаторов, если они написаны в виде отдельных файлов. В обоих случаях в обработку будут взяты одноименные стилевые файлы, js-файлы и картинки из папки `img/` блока.

Пример, подключающий 3 блока:

```
"blocks": {
  "page": [],
  "page-header": [],
  "page-footer": []
}
```

addCssBefore

Массив с дополнительными стилевыми файлами, которые будут взяты в компиляцию ПЕРЕД стилевыми файлами блоков.

Пример, берущий в компиляцию переменные, примеси, функции и один дополнительный файл из папки зависимостей (он будет преобразован в css-импорт, который при постпроцессинге ([postcss-import](#)) будет заменен на содержимое файла).

```
"addCssBefore": [
  "./src/scss/variables.scss",
  "./src/scss/mixins.scss",
  "./src/scss/functions.scss",
  "../../node_modules/owl.carousel/dist/assets/owl.carousel.css"
],
```

addCssAfter

Массив с дополнительными стилевыми файлами, которые будут взяты в компиляцию ПОСЛЕ стилевых файлов блоков.

```
"addCssAfter": [
  "./src/scss/print.scss"
],
```

singleCompiled

Массив стилевых файлов, которые будут скомпилированы независимо.

Пример: указанный файл будет скомпилирован в папку сборки как `blocks-library.css`

```
"singleCompiled": [
  "./src/scss/blocks-library.scss"
],
```

addJsBefore

Массив js-файлов, которые будут взяты в обработку (конкатенация/сжатие) ПЕРЕД js-файлами блоков.

Пример, добавляющий в список обрабатываемых js-файлов несколько зависимостей:

```
"addJsBefore": [
  "./node_modules/jquery/dist/jquery.min.js",
  "./node_modules/jquery-migrate/dist/jquery-migrate.min.js",
  "./node_modules/nouislider/distribute/nouislider.js"
],
```

addJsAfter

Массив js-файлов, которые будут взяты в обработку (конкатенация/сжатие) ПОСЛЕ js-файлов блоков.

Пример, добавляющий в конец списка обрабатываемых js-файлов глобальный скрипт.

```
"addJsAfter": [
  "./src/js/global-script.js"
],
```

addImages

Массив дополнительных изображений, добавляемый ПЕРЕД массивом изображений из блоков (внимание: при совпадении имен файлов, файлы из блоков имеют более высокий приоритет и затрут файлы из этого массива).

```
"addImages": [
  "./src/img/*.{jpg,jpeg,gif,png,svg,ico}"
],
```

copiedCss

Массив css-файлов, которые копируются в папку сборки, подпапку `css/`

copiedJs

Массив js-файлов, которые копируются в папку сборки, подпапку `js/`

ВНИМАНИЕ! Это JSON. Это строгий синтаксис, у последнего элемента в любом контексте не должно быть запятой в конце строки.

Пример секции в `./projectConfig.json`

```
{
  "blocks": {
    "page-header": [],
    "page-footer": [
      "__extra-element",
      "--extra-modifier"
    ]
  },
  "addCssBefore": [
    "./src/scss/variables.scss"
  ],
  "addCssAfter": [
    "./src/scss/print.scss"
  ],
  "singleCompiled": [],
  "addJsBefore": [
    "./node_modules/jquery/dist/jquery.min.js",
    "./node_modules/jquery-migrate/dist/jquery-migrate.min.js"
  ],
  "addJsAfter": [
    "./src/js/global-script.js"
  ],
  "addImages": [
    "./src/img/*.{jpg,jpeg,gif,png,svg}"
  ],
  "copiedCss": [],
  "copiedJs": [],
  "dirs": {
    "srcPath": "./src/",
    "buildPath": "./build/",
    "blocksDirName": "blocks"
  }
}
```

В результате в обработку будут взяты (в указанной последовательности):

```
css:
[ './src/scss/variables.scss',
  './src/blocks/page-header/page-header.scss',
  './src/blocks/page-footer/page-footer.scss',
  './src/blocks/page-footer/page-footer__extra-element.scss',
```

```

    './src/blocks/page-footer/page-footer--extra-modifier.scss',
    './src/scss/print.scss' ],
js:
[ './node_modules/jquery/dist/jquery.min.js',
  './node_modules/jquery-migrate/dist/jquery-migrate.min.js',
  './src/blocks/page-header/page-header.js',
  './src/blocks/page-footer/page-footer.js',
  './src/blocks/page-footer/page-footer__extra-element.js',
  './src/blocks/page-footer/page-footer--extra-modifier.js',
  './src/js/global-script.js' ],
img:
[ './src/img/*.{jpg,jpeg,gif,png,svg}',
  './src/blocks/page-header/img/*.{jpg,jpeg,gif,png,svg}',
  './src/blocks/page-footer/img/*.{jpg,jpeg,gif,png,svg}' ]

```

Удобное создание нового блока

Предусмотрена команда для быстрого создания файловой структуры нового блока.

```

# формат: node createBlock.js ИМЯБЛОКА [доп. расширения через пробел]
node createBlock.js block-1 # создаст папку блока, block-1.html, block-1.scss и подпапку img/ для этого блока
node createBlock.js block-2 js pug # создаст папку блока, block-2.html, block-2.scss, block-2.js, block-2.pug и подпа

```

Если блок уже существует, файлы не будут затёрты, но создадутся те файлы, которые ещё не существуют.

Назначение папок

```

build/      # Папка сборки, здесь работает сервер автообновлений.
src/        # Исходные файлы
_include/   # - фрагменты html для вставки на страницы
blocks/     # - блоки проекта
css/        # - можно положить добавочные css-файлы (нужно подключить в copiedCss, иначе игнорируются)
fonts/      # - можно положить шрифты проекта (будут автоматически скопированы в папку сборки)
img/        # - можно положить добавочные картинки (нужно подключить в addImages, иначе игнорируются)
js/         # - можно положить добавочные js-файлы (нужно подключить в addJsBefore, addJsAfter или copiedJs, иначе)
scss/       # - стили (style.scss скомпилируется, прочие нужно подключить в addCssBefore, addCssAfter или singleC
index.html  # - главная страница проекта
blocks-demo.html # - библиотека блоков

```

Комментирование для разработчиков

Для html-файлов можно использовать комментарии вида `<!--DEV Комментарий -->` — такие комментарии не попадут в собранный html.