Оглавление

Архитектура

Проект состоит из следующих файлов/папок:

```
build                           – готовая сборка
config                          – таски/конфиги сбощика
src                             - исходники
        components              - компоненты
                config          - конфигурация сайта
                icon            - компонент SVG иконки
                sprite          - компонент PNG спрайта
        fonts                   - шрифты
        icons                   - SVG иконки
        js                      - скрпиты
                core            - вспомогательные скрипты
                helpers         - полифилы
                main.js         - главный скрипт (точка входа)
        layouts                 - схемы страницы
                _default.pug    - схема по умолчанию
        pages
                index.pug       - Главная страница
        public                  - Публичная папка
                favicons        - фавиконки
                images          - изображения
                .htaccess       - настройки веб-сервера
        scss                    - стили
                helpers         - вспомогательные стили/переменные/миксины
                _common.scss    - главный файл пользовательских стилей
                _fonts.scss     - подключение шрифтов
                main.scss       - главный файл стилей (точка входа)
        sprite
tmp                             - Папка для временных файлов генерируемых сборщиком
tools                           - Вспомогательные инструменты для сборщика
.babelrc                        - конфиг babel
.browserlistrc                  - конфиг browserlist
.editorconfig                   - конфиг редактора
.gitignore
gulpfile.babel.js               - точка входа сборщика
package.json
postcss.config.js               - конфиг postcss
settings.js                     - настройка переменных (не завершено)
sw-precache-config.js           - конфиг сервис воркера
webpack.config.babel.js         - конфиг webpack
```

Иконки, спрайты, изображения

Иконки в формате SVG складываем в папку «src/icons». Из этих иконок сборщик генерирует спрайт.

Пример использования:

Например, мы положили в папку src/icons файл vk.svg

```
//Подключаем компонент иконки
include ../components/icon/icon
+icon(name="vk")
```

Аналогичным образом происходит работа со спрайтами PNG.

В папку src/sprite складываем иконки в формате PNG.

Пример использования:
Например мы положили в папку src/sprite файл error.png

```
//Подключаем компонент спрайта
include ../components/sprite/sprite
+sprite(name="error")
```

Все остальные изображения сказываются в папку src/public/assets/images.
Пример использования:
Например мы положили в папку src/public/assets/images файл preview.jpg

```
+b('image')(role="img" alt="Preview" src="assets/images/content/preview.png")
```

Стили

Для написания стилей используется препроцессор SASS. Главный файл находится в папке src/scss/ называется main.scss
Все вспомогательные файлы стилей, стили компонентов, миксины, переменные и т.д. подключаются в этом файле.
Пользовательские глобальные стили описываются в файле _common.scss.

Файлы начинающиеся с _ (например, _common.scss) не компилируются сборщиком в отдельный файл. Они используются для импорта .

Файл _fonts.scss используется для подключения шрифтов.
В папке helpers находятся вспомогательные переменные и миксины, упрощающие написание стилей.
_animation.scss – переменные для анимаций.
_colors.scss – переменные цветов.
_fonts.scss – миксин для подключения шрифтов.
_grid.scss – переменные и миксины для работы с сеткой.
_typography.scss – переменные для типографики.

Так же стили создаются в папке компонента. *Важно: В папке компонента находятся только стили для этого компонента.*

Если вы не знакомы с препроцессором SASS – настоятельно рекомендую вам изучить основы работы с данным инструментов. (https://sass-scss.ru/)

Скрипты

Для написания скриптов можно использовать ES6. Главным файлом (точкой входа) является src/js/main.js.

Скрпиты для компонентов размещать в папке компонента и описывать только логику компонента.
Компоненты можно описывать в виде плагина, используя вспомогательный файл Plugin.js
Пример плагина:

```
//Импортируем нужные инструменты
import Plugin, { init } from "@/modules/_utils/Plugin";


class Accordion extends Plugin {
  defaults() {
    return {
      //Опции по умолчанию, например
       itemsSelector: `[data-item]`
    };
  }

  init() {
    //Инициализация плагина
  }


  buildCache() {
       //Описываем внутренние переменные
    this.items = this.element.querySelectorAll(this.options.itemSelector)
  }

  bindEvents() {
    //Навешиваем события на нужные элементы
  }

  //Пользовательские функции

}

export default init(Accordion, "accordion");
```

Затем в main.js подключаем наш плагин

```
//Импортируем плагин
import Accordion from "../components/accordion/accordion.js";

//Вызываем плагин
Accordion(".accordion");
```

PUG

В качестве инструмента для работы с HTML используется шаблонизатор PUG, с подключенным модулем bempug. Верстать опираясь на методологию БЭМ (никакой отсебятины), и с использованием миксинов bempug.
Если незнакомы с методологией БЭМ - https://ru.bem.info/
Для знакомства с PUG - https://pugjs.org/api/getting-started.html
Для знакомства с BEMPUG - https://github.com/werty1001/bempug

Данные

Контентные данные можно выносить в отдельный yml файл.
Например, у нас есть компонент main-menu. Создадим в папке компонента папку data, а в ней файл main-menu.yml (yml-файл именуем так же как и компонент).

```yaml
//main-menu.yml
---
  mainMenu:
    default:
      -
        title: Главная
        url: main
      -
        title: О проекте
        url: about
      -
        title: Документы
        url: documents
      -
        title: Техподдержка
        url: support
      -
        title: Контакты
        url: contacts
```

Теперь мы можем использовать этот контент в верстке

```
//main-menu.pug
mixin main-menu(data)
    +b('main-menu', {t: 'nav'})&attributes(attributes)
        +e('list', {t: 'ul'})
            each item in data
                +e('item')
                    +e('link', {t: 'a'})(href=`${item.url}.html`)= item.title
```

Вызывая миксин main-menu в качестве параметров передаем ему переменную mainMenu созданную в yml файле.

```
//Например, вызовем наш миксин в каком-нибудь header.pug
include ../main-menu/main-menu

+main-menu(mainMenu.default)
```

Шрифты

Шрифты расположены в папке src/fonts. При сборке проекта они переносятся в папку build.

Public

В публичной папке находятся файлы которые должны быть помещены в финальную сборку. Структура папок сохраняется.

Инструменты

Для создания компонента можно использовать команду

---

npm run cc <имя компонента> <параметры>

если выполнить эту команду без параметров, то в папке components будет создана папка «Имя компонента», а в ней два файла «Имя компонента.pug» и «Имя компонента.scss».
В качестве параметра можно указать yml или js или yml js. Указание данных параметров позволит создать дополнительно yml и js файл.

//TODO

- Бэкенд заглушка
- Settings.js