**Препроцессор SASS (для CSS)**

Сайт для компиляции <https://scout-app.io/>

Либо плагин для VS кода Live Sass Compiler

Настройка в VS коде (в файле settings.json):

"liveSassCompile.settings.formats": [

      {

        "format": "compressed",

        "extensionName": ".min.css",

        "savePath": "~/../css/"

      }

    ],

    "liveSassCompile.settings.autoprefix": [

      "> 1%",

      "last 2 versions"

    ]

**Flex боксы**

У родителя выравниваемых элементов

**display: flex;**

По вертикали **justify-content: center;**

По горизонтали **align-items: center;**

**Выравнивание по середине без Flex**

При **position: absolute;**

**left: 50%;**

**transform: translate(-50%);**

**Скорость анимации CSS**

**transition: 0.5s all;**

где **all** – это те свойства **css**, на которые будет действовать данный **transition**

**:after и :before**

**Пример применения :before для создания плавного подчеркивания под ссылкой в меню**

.red:before {

content: ‘ ’;

display: block;

height: 2px;

width: 2px;

background-color: #000;

transition: 0.5s all;

}

.red:hover:before {

width: 100%;

}

**Ресурс для сборки собственных icon-шрифтов**

[**https://icomoon.io/app/#/select**](https://icomoon.io/app/#/select) **- сборка шрифтов в свой файл**

[**https://fontawesome.com/icons?d=gallery&p=2**](https://fontawesome.com/icons?d=gallery&p=2) **– иконки**

[**https://seeklogo.com/**](https://seeklogo.com/) **- менее удобный ресурс с иконками, но их тут больше**

**Работа с определенными элементами родителя с одинаковыми классами**

**.name\_div > div:nth-child(3) {**

**Display: flex;**

**Justify-content: flex-end;**

**}**

Здесь мы прижимаем третий элемент **div**-а внутри блока **.name\_div** к левому краю. ( **> div** означает, что мы применяет это свойство ко всем div внутри **.name\_div**)

**Стиль верстки Perfect Pixel**

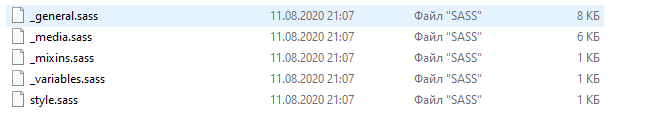
Данный стиль подразумевает 100% соответствие сайта макету, пиксель в пиксель.

Для проверки соответствия стоит использовать дополнение для Chrome “PerfectPixel”, которое позволяет наложить на страницу скриншот и посмотреть, какие элементы не попадает в макет.

<https://chrome.google.com/webstore/detail/perfectpixel-by-welldonec/dkaagdgjmgdmbnecmcefdhjekcoceebi>

**Разделение Препроцессора SASS**

Создаем в той же папке, где и находится **style.sass** файлы с названиями “\_name.sass”



Затем подключаем их в файле **style.sass**, не используя в начале названий “\_”, но это не обязательно, т.к. эта “\_” служит напоминанием Вам и другим разработчикам, что данный файл куда-то импортируется.

@import "variables"

@import "mixins"

@import "general"

@import "media"

**Отступы в bootstrap**

Для создания отступа на определенное количество колонок слева, можно добавить класс, к примеру, **.offset-md-2**, тогда на экранах размером **md** перед данным блоком появится отступ в две колонки

**Грамотная адаптация через SASS**

Создадим файл с названием \_media.sass и запишем в него поочередно каждую максимальную ширину экрана со всеми стилями и их изменениями, которые нужно применить к элементам, для корректного отображения

@media (max-width: 1200px)

    .menu

        &\_item

            padding-right: 20px

        &\_link

            font-size: 13px

    .promo

        &\_header

            font-size: 50px

    .reasons

        &\_block

            align-items: flex-start

        &\_descr

            width: 300px

@media (max-width: 991px)

    .title

        font-size: 28px

    .menu

        &\_item

            padding-right: 8px

        &\_link

            font-size: 10px

@media (max-width: 767px)

    .title

        margin-top: 25px

    .menu

        position: fixed

        flex-direction: column

        justify-content: center

        align-items: center

        top: 0

и т.д.

**Изменения гамбургера на крестик при клике**

.hamburger

        display: block

        position: absolute

        right: 10px

        top: 50%

        transform: translateY(-50%)

        width: 26px

        height: 12px

        span

            display: block

            height: 2px

            width: 100%

            background-color: #fff

            margin-bottom: 2px

            transition: 0.5s all

        &\_active

            span

                margin-bottom: -2px

                &:nth-child(1)

                    transform: translateY(3px) rotate(-45deg)

                &:nth-child(2)

                    display: none

                &:nth-child(3)

                    transform: translateY(3px) rotate(45deg)

**Git и работа с ним**

1. Уставить Git <https://git-scm.com/download/win>
2. Локально настроить Git через консоль (терминал)

Это можно сделать непосредственно в VS Code через терминал

* 1. логинимся локально

git config –local user.name “Ilya”

git config –local user.email [example@gmail.com](mailto:example@gmail.com)

* 1. проверяем через команду

git status

1. Добавляем все наши файлы в репозиторий (в index)

Git add -A

-A означает ВСЕ файлы, чтобы добавить определенный файл, пишем

Git add name.css

Или все файлы с расширением .css

Git add \*.css

1. Создаем контрольную точку

Git commit -a -m”описание”

Где -a все файлы, -m описание контрольной точки

1. Создаем следующую контрольную точку (commit)

Git add -A

Git add -a -m”second”

1. Просмотр изменений

Git log

1. Заливаем репозиторий на GitHub

Логинимся на GitHub

Создаем новый репозиторий

Копируем код и вставляем в терминал

1. Для отправки наших файлов на удаленный репозиторий прописываем команду (или копируем оттуда же)

Git remote add origin https://github.com/name/progect.git

Git push -u origin master

Где master – название нашей основной ветки, это название задается стандартно, но мы можем его поменять, прописав git branch -M name.

Так же, в наши дня из-за толерантности GitHub переименовывает главную ветку из master в main

- u означает, что в дальнейшем все пуши будут происходить туда же автоматически

1. Для дальнейшей отправки в удаленный репозиторий повторяем цепочку

Git add -A

Git commit -a -m”new”

И прописываем уже простую команду

Git push

И при запуске этой команды все отправится в репозиторий, т.к. до этого мы прописали ключ -u

**Работа со своими проектами на разных компах через Git**

1. На GitHab через окно скачивания репозитория берем ссылку на наш проект
2. В терминале выходим на одну папку назад через команду cd ..
3. Зайдем в заранее созданную папку, к примеру work

Cd work

1. Прописываем команду для клонирования нашего проекта на компьютер

Git clone https://(скопированная ссылка).git it progect\_2

Где it progect\_2 – название папки, в которой у нас все это будет храниться

1. Также пушим все наши изменения в репозиторий, при чем, если мы скачиваем уже готовый репозиторий, то все настройки пуша сохраняются, т.к. они прописаны в скрытой папке репозитория.
2. Для того, чтобы продолжить работать с измененным проектом на другом компьютере, где у нас уже есть его старая версия, то мы заходим в нее и прописываем в терминале команду

Git pull

Она обновляет наш проект до его последней версии, которая есть на GitHub

1. Если кто-то внес изменения в репозиторий, а мы об это не знаем и пытаемся запушить нашу новую версию, то произойдет ошибка, после чего на необходимо запулить новую версию к нам и произойдет переход в режим мерча (слияния), который создаст файл READ.md, в котором мы можем прописать описание мерча прямо в терминале, после описания нужно нажать ctrl+C два раза, затем :wq! и нажать Enter

Для удобного просмотра изменения в репозиториях GitKraken

<https://www.gitkraken.com/>

Также в Git существует файл .gitignore, в который записываются файлы, которые не нужно загружать в репозиторий

**Работа с Gulp**

1. Для правильной работы нужно переставить политику безопасности из Restricted в Unrestricted

Для этого откроем PowerShell от админа и пропишем Set-ExecutionPolicy Unrestricted, затем y для согласия на отключение

После запуска Gulp нужно прописать команду на обратное изменение политики безопасности в том же PowerShell: Set-ExecutionPolicy Restricted

1. Для установки Gulp нужно:
   1. Скачать и установить <https://nodejs.org/en/>
   2. Установить сам Gulp в проекте сначала глобально (единожды на ПК), а затем локально для данного проекта (для каждого проекта каждый раз):

В терминале прописываем: **npm init** и соглашаемся ентарами со всеми вопросами

Затем устанавливаем gulp глобально

npm install gulp -g

Затем локально в этот проект

npm install gulp-cli --save-dev

После чего устанавливаем npm install gulp-cli -g

npm install gulp-cli --save-dev

После чего устанавливаем все нужные нам пакеты:

npm install -g browser-sync

npm install browser-sync --save-dev

npm i gulp-sass -g

npm i gulp-sass --save-dev

npm i gulp-autoprefixer -g

npm i gulp-autoprefixer --save-dev

npm i gulp-clean-css -g

npm i gulp-clean-css --save-dev

npm i gulp-rename -g

npm i gulp-rename --save-dev

Теперь нужно создать файл gulpfile.js и прописать в нем все настройки наших пакетов.

Благо, для упомянутых выше пакетов у нас есть файл с настройками

<https://drive.google.com/file/d/1lBAXcbteahnUQJfaYEk78Km_I4BgbVIg/view>

при этом в файл package.json нужно добавить конструкцию

"browserslist": [

    "last 1 version",

    "> 1%",

    "maintained node versions",

    "not dead"

  ],

И выглядеть это должно так

 "gulp-cli": "^2.3.0",

    "gulp-rename": "^2.0.0",

    "gulp-sass": "^4.1.0"

  },

  "browserslist": [

    "last 1 version",

    "> 1%",

    "maintained node versions",

    "not dead"

  ]

}

Эта конструкция отвечает за настройку плагина, отвечающего за авто подстановку префиксов для работы разных браузеров