**12.3 Версии, скрипты. Semantic versioning**

<https://lms.skillfactory.ru/courses/course-v1:Skillfactory+FR+2020/courseware/4277fd5e8f184b7ebf2eb5b830e09fdd/60f23eb56d1b4eacace9c2f7c291b31b/2?activate_block_id=block-v1%3ASkillfactory%2BFR%2B2020%2Btype%40vertical%2Bblock%40d71188c35ea8440099ff477112f55135>

## ****Дополнительно****

[Руководство по семантическому версионированию и FAQ](https://semver.org/lang/ru/)

<https://semver.org/lang/ru/>

## 12.4 CSS-препроцессоры

## ****Зачем нужны препроцессоры****

Написание CSS-кода, помимо огромного количества плюсов, содержит в себе и минусы. Чем больше CSS-часть кода, тем сложней этот код поддерживать и конфигурировать.

CSS-препроцессоры вроде [SASS](https://sass-lang.com/) и [LESS](http://lesscss.org/) созданы, чтобы предварительно обрабатывать .sass (.scss) и .less файлы, получая на выходе оптимизированный CSS-файл, который затем и встраивается в HTML.

[SASS](https://sass-lang.com/) https://sass-lang.com/ ; [LESS](http://lesscss.org/) https://lesscss.org/

**Препроцессор** — это программа, которая принимает данные на входе, обрабатывает их и выдаёт данные для другой программы.

Рассмотрим, как она работает, на примере.

<https://lms.skillfactory.ru/courses/course-v1:Skillfactory+FR+2020/courseware/4277fd5e8f184b7ebf2eb5b830e09fdd/60f23eb56d1b4eacace9c2f7c291b31b/2?activate_block_id=block-v1%3ASkillfactory%2BFR%2B2020%2Btype%40vertical%2Bblock%40d71188c35ea8440099ff477112f55135>

Cимволы «**\_**» и «**-**» в названиях переменных взаимозаменяемы. Например, использование переменной **$table-border** будет эквивалентно использованию **$table\_border**.

Следует отметить, что переменные можно объявлять в любом месте файла, но ДО использования.

### Задание 12.4.1

1/1 point (graded)

Выберите правильное объявление переменной в SASS:



Верно

## ****Как использовать @import****

https://lms.skillfactory.ru/courses/course-v1:Skillfactory+FR+2020/courseware/4277fd5e8f184b7ebf2eb5b830e09fdd/60f23eb56d1b4eacace9c2f7c291b31b/2?activate\_block\_id=block-v1%3ASkillfactory%2BFR%2B2020%2Btype%40vertical%2Bblock%40d71188c35ea8440099ff477112f55135

Одно из больших преимуществ SASS над CSS состоит в лёгкости его настройки. В пару кликов можно автоматически добавлять префиксы в выходной CSS-файл, минифицировать его, собирать отдельные файлы для разных страниц путём объединения необходимых конкретно для каждой страницы деталей в виде отдельных файлов, отвечающих за свои собственные стили (отдельно стили для списков, таблиц, повторяющихся элементов по типу header и footer).

Одна из самых удобных функциональных возможностей в SASS (относится и к LESS) — **вложенные правила**. Разработчики составляют сложные селекторы, чтоб обезопасить свой CSS-код от действия вне необходимого элемента.

Вместо составления таких селекторов в SASS пользуются вложенными правилами в формате:

<селектор> {

/\* стили селектора \*/

<вложенный селектор> {

/\* стили вложенного селектора \*/

}

}

Это трансформируется в CSS вида:

<селектор> { /\* стили селектора \*/ }

<селектор> <вложенный селектор> { /\* стили вложенного селектора \*/ }

Если при составлении вложенного селектора необходимо обратиться к родительскому селектору (чтоб навесить стили hover, указать дополнительные селекторы родителя), используют символ **&**.

SASS-код:

div {

a {

**&:hover**, **&:focus**, **&:active** {

text-decoration: **none**;

}

}

**&.hidden** {

display: **none**;

**&**-as-opacity {

opacity: **0**;

}

}

}

Трансформируется в следующий CSS:

div a**:hover**, div a**:focus**, div a**:active** {

**text-decoration**: **none**;

}

div**.hidden** {

**display**: **none**;

}

div**.hidden-as-opacity** {

**opacity**: **0**;

}

### Задание 12.4.2

1/1 point (graded)

Выберите правильный импорт SCSS-файла с переменными:



верно

SASS поддерживает стандартные арифметические операции (+, -, \*, /, % (остаток по модулю)). При этом не разрешаются операции над несовместимыми типами данных, например, px + em.

**Пример**:

p {

margin: **3px** + **4px** **auto**;

}

Компилируется в ...

p {

margin: **7px** **auto**;

}

Помимо этого, можно использовать **интерполяцию** (подстановка переменных SASS в селекторах и в названиях свойств) через синтаксис **#{}**.

**Пример**:

$value: **8**;

p**:before** {

content: "№ #{$value + **10**}";

}

## img****Дополнительно****

[ES6: Интерполяция](http://jsraccoon.ru/es6-interpolation)

<http://jsraccoon.ru/es6-interpolation>

**Миксины**(от англ. mixins — «примеси») — стили, которые могут быть переиспользованы во избежание применения дополнительных классов. Также они могут принимать аргументы, что обеспечивает широкую функциональность миксинов при минимальном их количестве.

## Объявление миксина:

@mixin <название миксина> <(перечисление аргументов через запятую, если необходимо)> {

/\* CSS \*/

}

## Использование миксина:

<селектор> {

@include <название миксина> <(указание аргументов в том же порядке, который описан в миксине)>;

/\* собственный CSS \*/

}

**Пример:**

**@mixin centered-block** ($width, $background) {

width: $width;

margin: **0** **auto**;

background-color: $background;

}

**.container** {

**@include centered-block**(**50%**, **#dadada**);

min-width: **400px**;

}

Документация SASS [здесь](https://sass-scss.ru/documentation/).

https://sass-scss.ru/documentation/

Описание пакета node\_sass [здесь](https://www.npmjs.com/package/node-sass).

<https://www.npmjs.com/package/node-sass>

Какой миксин объявлен правильно?

@mixin border($border\_width) { border: $border\_width solid black; }

## 12.5 Методология БЭМ

<https://lms.skillfactory.ru/courses/course-v1:Skillfactory+FR+2020/courseware/4277fd5e8f184b7ebf2eb5b830e09fdd/60f23eb56d1b4eacace9c2f7c291b31b/2?activate_block_id=block-v1%3ASkillfactory%2BFR%2B2020%2Btype%40vertical%2Bblock%40d71188c35ea8440099ff477112f55135>

С развитием веб-разработки всё большее значение приобретает работа над оптимизацией структуры проекта, что мы и видим по уже пройденным темам. Изначально правильно организованная структура уменьшает порог вхождения разработчика в проект (время, затраченное на достижение определённых умений для работы в нём).

Большинство проблем в организации *HTML*+*CSS* кода связаны с тесным связыванием элементов в единую сеть. Чем больше проект, тем больше обычно нужно времени на разработку новых элементов, чтобы встроить их в систему. Также усложняется поддержка кода из-за хаотично названных элементов. Методологии (подходы) к веб-разработке призваны минимизировать проблему неорганизованности кода. Одной из таких методологий является *BEM*.

**BEM** расшифровывается как Блок (*Block*), Элемент (*Element*), Модификатор (*Modifier*).

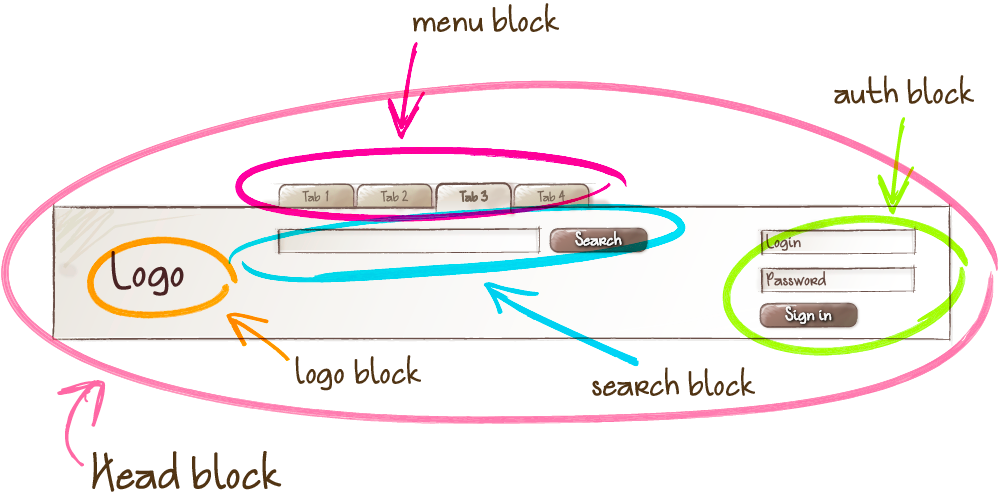
В основе *BEM* лежит подход разделения структуры страницы на независимые блоки с элементами и модификаторами внутри. Разделение происходит посредством правильного именования элементов страницы классами.

* + **Блок** — независимый компонент для повторного использования;
  + **Элемент** — составная часть блока (элемент меню в блоке меню, к примеру);
  + **Модификатор** — указание свойства для блока или элемента (примеры: активный элемент меню, *input*, доступный только для чтения).

Документация [здесь](https://ru.bem.info/methodology/quick-start/) (rus) и [здесь](http://getbem.com/introduction/) (eng).

<https://ru.bem.info/methodology/quick-start/>

## Расположение блоков

Источник: [en.bem.info](https://en.bem.info/methodology/key-concepts/" \t "_blank)

## Синтаксис

При именовании классов по BEM необходимо составное название блока/элемента/модификатора разделять с помощью «-».

Элемент принадлежит блоку и записывается как:

<имя-блока>\_\_<название-элемента>

Модификатор описывает состояние блока или элемента и именуется в формате:

<имя-блока>\_<название-модификатора>

или же

<название элемента>\_<название-модификатора>

**Пример вёрстки по BEM:**

<form class=”search-form”> <!-- Блок search-form-->

<h2 class=”search-form\_\_title”>Search form</h2> <!-- Элемент search-form\_\_title -->

<input class=”search-form\_\_input-field” placeholder=”Please type here..” />

<input id=”terms” class=”search-form\_\_terms-checkbox search-form\_\_terms-checked” type=”checkbox” checked> <!-- Элемент search-form\_\_terms-checkbox с модификатором search-form\_\_terms-checked -->

<label for=”terms” class=”search-form\_\_terms-label”>I accept Terms and Conditions</label> <!-- Элемент search-form\_\_terms-label -->

</form>

О файловой структуре [в документации](https://ru.bem.info/methodology/quick-start/#%D1%84%D0%B0%D0%B9%D0%BB%D0%BE%D0%B2%D0%B0%D1%8F-%D1%81%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%B0).

<https://ru.bem.info/methodology/quick-start/#%D1%84%D0%B0%D0%B9%D0%BB%D0%BE%D0%B2%D0%B0%D1%8F-%D1%81%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%B0>

Выберите правильное название класса для блока:



верно

Отправить

В некоторых задачах доступны следующие действия: сохранение, сброс, показ подсказки или ответа. Соответствующие кнопки расположены рядом с кнопкой «Отправить».

Показать Ответ

### Задание 12.5.2

1/1 point (graded)

Выберите правильное название класса для элемента:



верно

Отправить

В некоторых задачах доступны следующие действия: сохранение, сброс, показ подсказки или ответа. Соответствующие кнопки расположены рядом с кнопкой «Отправить».

Показать Ответ

### Задание 12.5.3

1/1 point (graded)

Выберите правильное название класса для модификатора:



Верно

## 12.6 Итоги

Ещё один модуль позади! Вы отлично справляетесь, и вот что вы освоили на этой неделе:

* + Изучили, как работать с package.json, поднимая на своём устройстве проект или создавая новый через npm или yarn;
  + Научились правильно определять версии зависимостей (пакетов), чтобы избежать проблем с обратной совместимостью при выходе новых версий;
  + Научились добавлять/удалять зависимости (пакеты) из проекта;
  + Знаете, как работать с препроцессорами на примере *SASS*, создавать структуру *SASS*-файлов и компилировать *SASS* в *CSS* с помощью npm.
  + Освоили базовые принципы методологии *BEM* для улучшения качества кода в процессе веб-разработки.