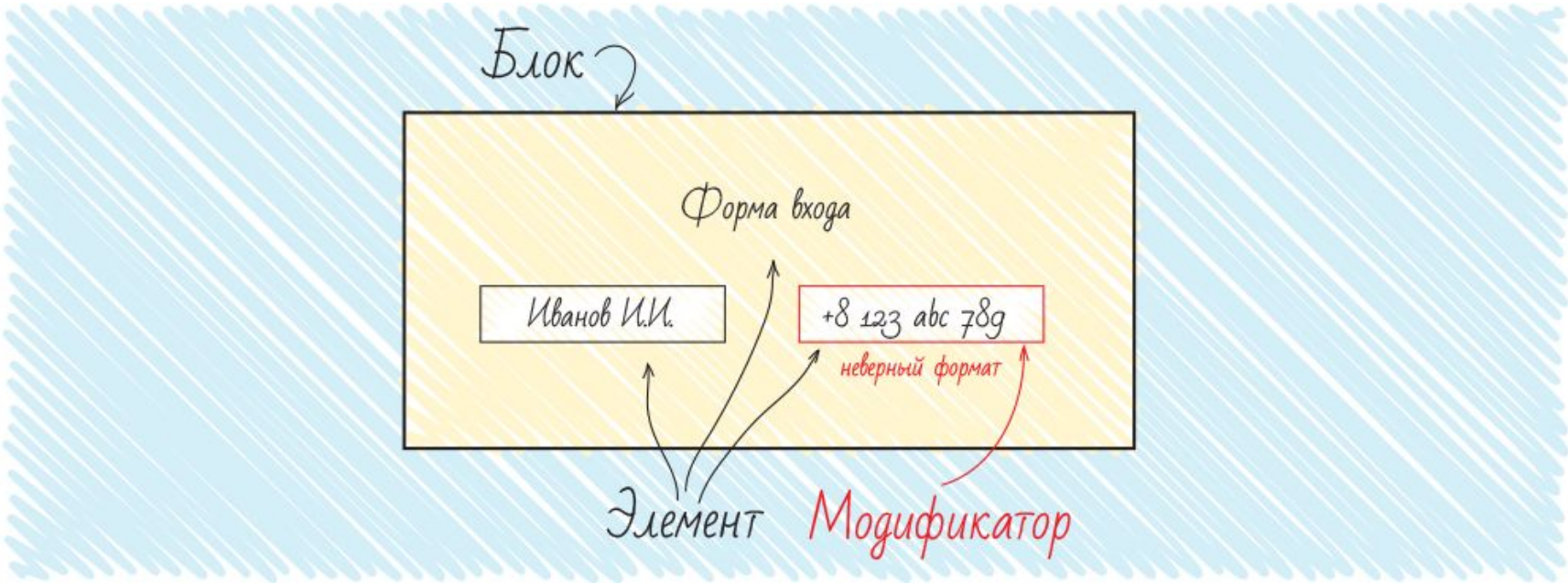
Разработка верстки. Архитектура CSS.

Способы организации кода

1. БЭМ
2. OOCSS
3. SMACSS
4. Atomic CSS
5. MCSS
6. AMCSS
7. FUN

БЭМ


Блок
Форма входа
Иванов И.И.
+8 123 abc 78g
неверный формат
Элемент
Модификатор

БЭМ - понятие

БЭМ (Блок, Элемент, Модификатор) — компонентный подход к веб-разработке. В его основе лежит принцип разделения интерфейса на независимые блоки. Он позволяет легко и быстро разрабатывать интерфейсы любой сложности и повторно использовать существующий код, избегая «Copy-Paste»

Блок

Функционально независимый компонент страницы, который может быть повторно использован. В HTML блоки представлены атрибутом class .

Блок - особенности

Название блока характеризует смысл («что это?» — «меню»: menu , «кнопка»: button), а не состояние («какой, как выглядит?» — «красный»: red , «большой»: big).

Верно:

Ошибка:

Блок - особенности

Блок не должен влиять на свое окружение, т. е. блоку не следует задавать внешнюю геометрию (в виде отступов, границ, влияющих на размеры) и позиционирование.

В CSS по БЭМ также не рекомендуется использовать селекторы по тегам или id .

Принцип работы с блоками

- Блоки можно вкладывать друг в друга.
- Допустима любая вложенность блоков

```
<header>
    <div class="logo"></div>
    <form class="search-form"></div>
</header>
```

Элемент

Составная часть блока, которая не может использоваться в отрыве от него.

```
<form class=“search-form”></div>
    <input class=“search-form__input”>
    <button class=“search-form__button”>найти</button>
</form>
```

Элемент – Особенности

- Название элемента характеризует смысл («что это?» — «пункт»: item , «текст»: text), а не состояние («какой, как выглядит?» — «красный»: red , «большой»: big).
- Структура полного имени элемента соответствует схеме:

  имя-блока__имя-элемента .

Имя элемента отделяется от имени блока двумя подчеркиваниями (__).

Принципы работы с элементами

1. Вложенность
2. Принадлежность
3. Необязательность

Вложенность

- Элементы можно вкладывать друг в друга.
- Допустима любая вложенность элементов.
- Элемент — всегда часть блока, а не другого элемента. Это означает, что в названии элементов нельзя прописывать иерархию вида block__elem1__elem2 .

Иерархия - пример

```
<form class="search-form">
    <div class="search-form__content">
     <input class="search-form__input">
     <button class="search-form__button">Найти</button>
    </div>
</form>
<form class="search-form">
    <div class="search-form__content">
        <input class="search-form__content__input">
        <button class="search-form__content__button">Найти</button>
     </div>
</form>
```

Вложенность - пример

```
<div class="block">
    <div class="block__elem1">
        <div class="block__elem2">
            <div class="block__elem3">
            </div>
        </div>
    </div>
</div>
```

Принадлежность

Элемент — всегда часть блока и не должен использоваться отдельно от него.

Необязательность

Элемент — необязательный компонент блока. Не у всех блоков должны быть элементы.

Блок или элемент

1. Блок
   Если фрагмент кода может использоваться повторно и не зависит от реализации других компонентов страницы.
2. Элемент

   Если фрагмент кода не может использоваться самостоятельно, без родительской сущности (блока). Исключение составляют элементы, реализация которых для упрощения разработки требует разделения на более мелкие части — подэлементы

Модификатор

Сущность, определяющая внешний вид, состояние или поведение блока либо элемента

```
<form class="search-form search-form_focused">
    <input class="search-form__input">      <button
class="search-form__button
search-form__button_disabled">Найти</button>
</form>
```

Модификатор - особенности

- Название модификатора характеризует
  - внешний вид («какой размер?», «какая тема?» и т. п. — «размер»: size_s , «тема»: theme_islands),
  - состояние («чем отличается от прочих?» — «отключен»: disabled , «фокусированный»: focused)
  - поведение («как ведет себя?», «как взаимодействует с пользователем?» — «направление»: directions_left-top).
- Имя модификатора отделяется от имени блока или элемента одним подчеркиванием (_).

Типы модификаторов

1. Булевый

Используют, когда важно только наличие или отсутствие модификатора, а его значение несущественно.

Например, «отключен»: disabled . Считается, что при наличии булевого модификатора у сущности его значение равно true.

Структура полного имени модификатора соответствует схеме:

имя-блока_имя-модификатора ;

имя-блока__имя-элемента_имя-модификатора.

Типы модификаторов

2. Ключ-значение

Используют, когда важно значение модификатора.

Например, «меню с темой оформления islands »: menu_theme_islands .

Структура полного имени модификатора соответствует схеме:

имя-блока_имя-модификатора_значение-модификатора ;

имя-блока__имя-элемента_имя-модификатора_значение-модификатора .

Принципы работы с модификаторами

1. Модификатор нельзя использовать самостоятельно

С точки зрения БЭМ-методологии модификатор не может использоваться в отрыве от модифицируемого блока или элемента. Модификатор должен изменять вид, поведение или состояние сущности, а не заменять ее.

Пример

```
<form class="search-form search-form_theme_islands">
    <input class="search-form__input">
    <button class="search-form__button">Найти</button>
</form>

<form class="search-form_theme_islands">
    <input class="search-form__input">
    <button class="search-form__button">Найти</button>
</form>
```

Принципы работы с модификаторами

2. Микс

Прием, позволяющий использовать разные БЭМ-сущности на одном DOM-узле.

Миксы позволяют:

1. совмещать поведение и стили нескольких сущностей без дублирования кода;
2. создавать семантически новые компоненты интерфейса на основе имеющихся.

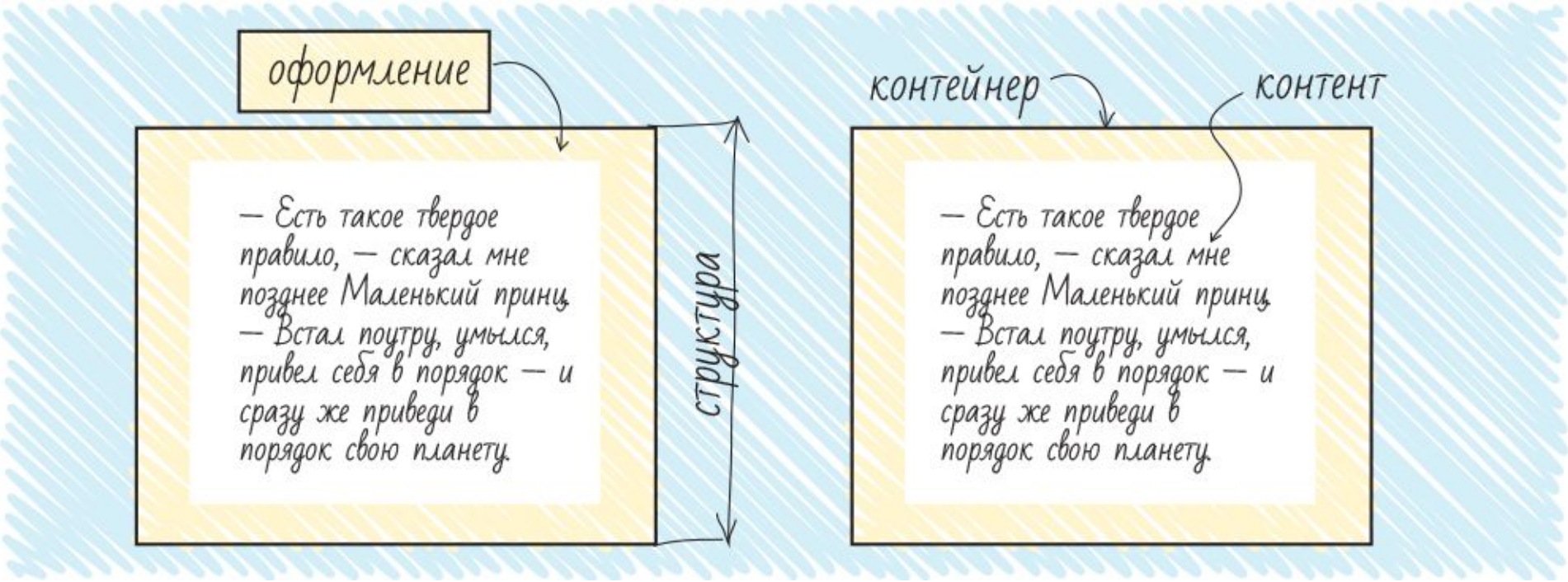
Пример

```
<div class="header">
    <div class="search-form header__search-form">
        ....
    </div>
</div>
```

БЭМ +/-

- избавление от непредсказуемых последствий каскада
- изолирование отдельных модулей друг от друга
- Длинные названия классов
- Избыточность различных классов
- Избыточность кода для одноразовых элементов

OOCSS


оформление
— Есть такое твердое правило, — сказал мне позднее Маленький принц. — Встал поутру, умылся, привел себя в порядок — и сразу же приведи в порядок свою планету.
структура
контейнер
контент
— Есть такое твердое правило, — сказал мне позднее Маленький принц. — Встал поутру, умылся, привел себя в порядок — и сразу же приведи в порядок свою планету.

OOCSS - понятие

OOCSS означает объектно-ориентированный CSS (Object-Oriented CSS). В этот подход заложены две основные идеи:

- Разделение структуры и оформления
- Разделение контейнера и контента (содержимого)

Разделение структуры и оформления

Характеристики выделяются в модули на основе классов, **они становятся многократного пользования** и могут с тем же результатом применяться к любому элементу.

Разделение контейнера и контента

Независимость наших стилей от любого элемента-контейнера. Это означает, что их затем можно **повторно применять где угодно** в документе независимо от структурного контекста.

Рекомендации к написанию кода

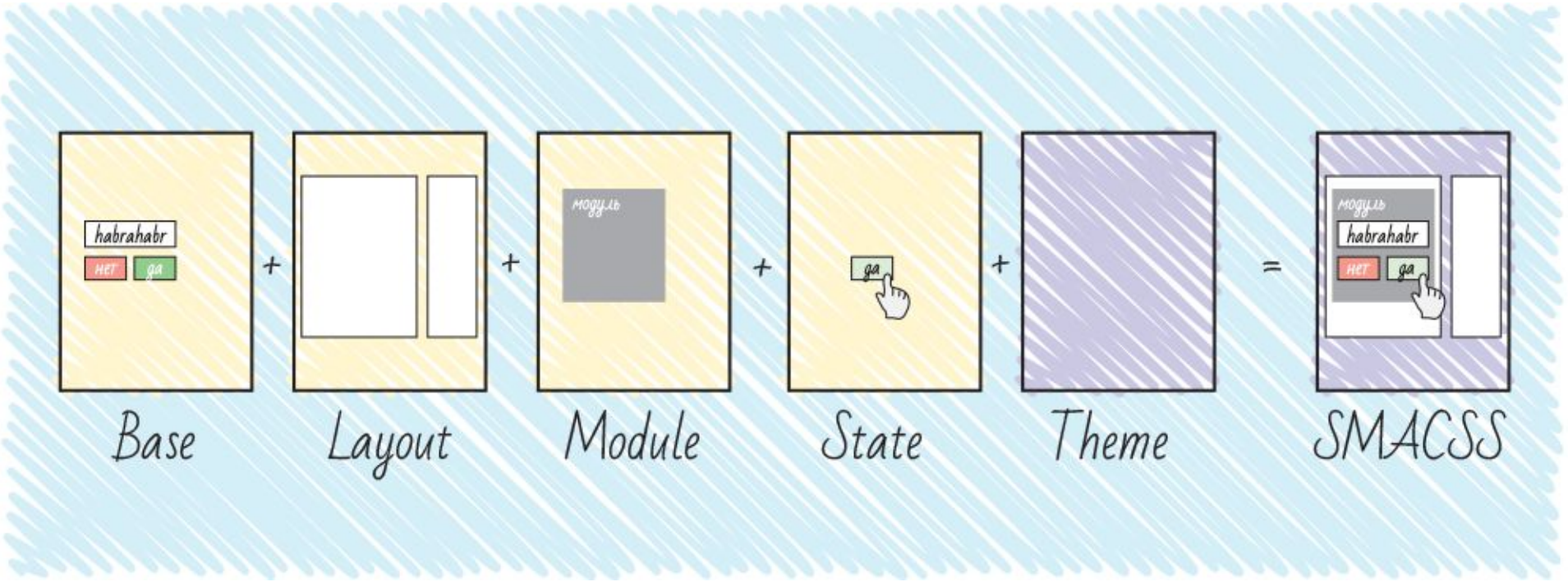
1. Избегайте селектора-потомка
2. Избегайте ID в качестве ловушек-прехватчиков стилей
3. Избегайте прикрепления классов к элементам в своей таблице стилей
4. За исключение некоторых редких случаев избегайте использования !important
5. Применяйте ***CSS Lint*** для проверки своего CSS Применяйте сетки CSS

OOCSS +/-

- уменьшение количества кода за счет повторного его использования

- достаточно сложная поддержка: при изменении стиля конкретного элемента скорее всего придется менять не только CSS (т.к. большинство классов общие), но и добавлять классы в разметку.

SMACSS


habrahabr
нет
да
+
+
модуль
+
да
+
=
модуль
habrahabr
нет
да
Base
Layout
Module
State
Theme
SMACSS

SMACSS - понятие

SMACSS расшифровывается как «масштабируемая и модульная архитектура для CSS» (Scalable and Modular Architecture for CSS).

Основная цель подхода — уменьшение количества кода и на упрощение поддержки кода.

SMACSS - структура

1. **Base rules**

   базовые стили. Это стили основных элементов сайта — body, input, button, ul, ol и т. п. В этой секции используются в основном селекторы тэгов и атрибутов, классы — в исключительных случаях (например, если у вас стилизованные JavaScript'ом селекты);

SMACSS - структура

2. **Layout rules**

   стили макета. Здесь находятся стили глобальных элементов размеры шапки, футера, сайдбара и т.п. Джонатан предлагает использовать здесь id в селекторах, так как эти элементы не будут встречаться более 1 раза на странице.

SMACSS - структура

3. Modules rules

стили модулей, то есть блоков, которые могут использоваться несколько раз на одной странице. Для классов модулей не рекомендуется использовать id и селекторы тэгов (для многократного использования и независимости от контекста соответственно).

SMACSS - структура

4. **State rules**

   стили состояния. В этом разделе прописываются различные состояния модулей и скелета сайта. Это единственный раздел, в котором допустимо использование ключевого слова «!important».

SMACSS - структура

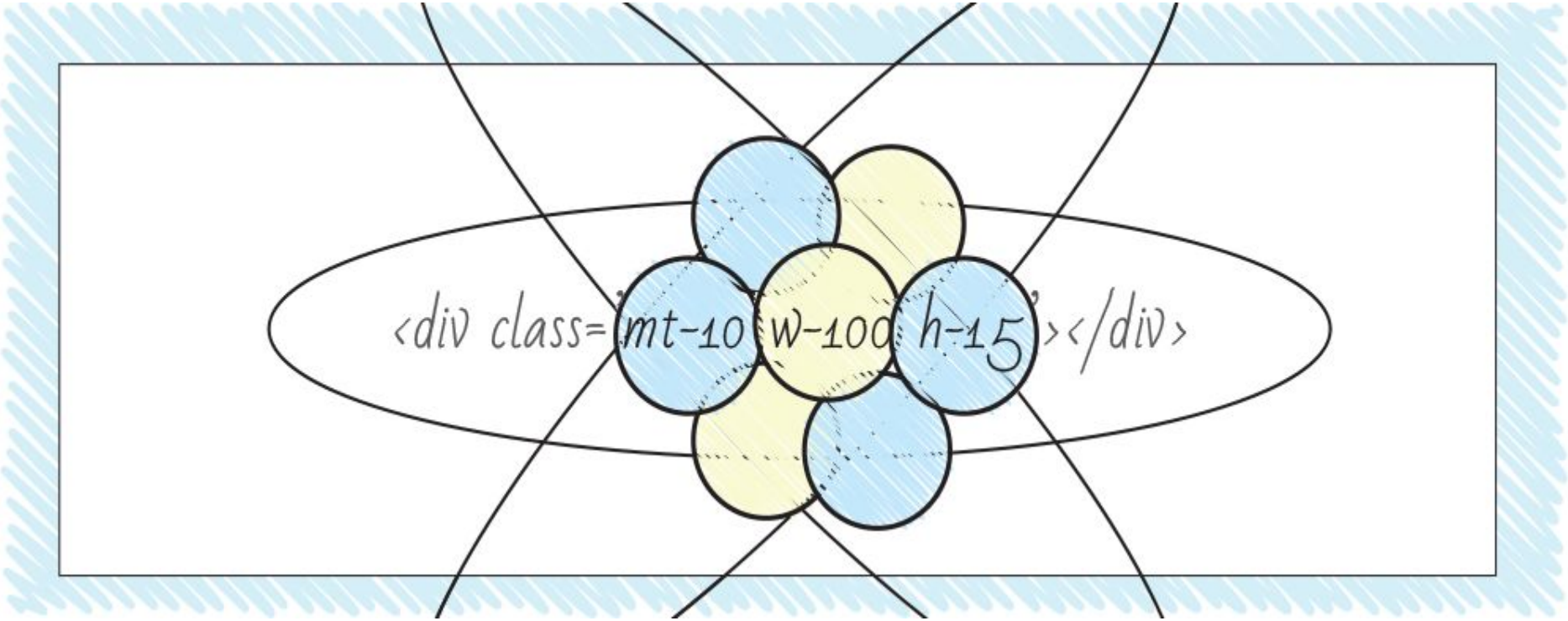
5. Theme rules

оформление. Здесь описываются стили оформлений, которые со временем, возможно, нужно будет заменить (так удобно делать, например, новогоднее оформление; для html-тем, выставленных на продажу такие стили позволяют переключать цветовую гамму и т.п.).

SMACSS - Namespaces

Рекомендуется вводить неймспейсы для классов, принадлежащих к определенной группе, а также использовать отдельный неймспейс для классов, используемых в JavaScript.

Atomic CSS



Atomic CSS - понятие

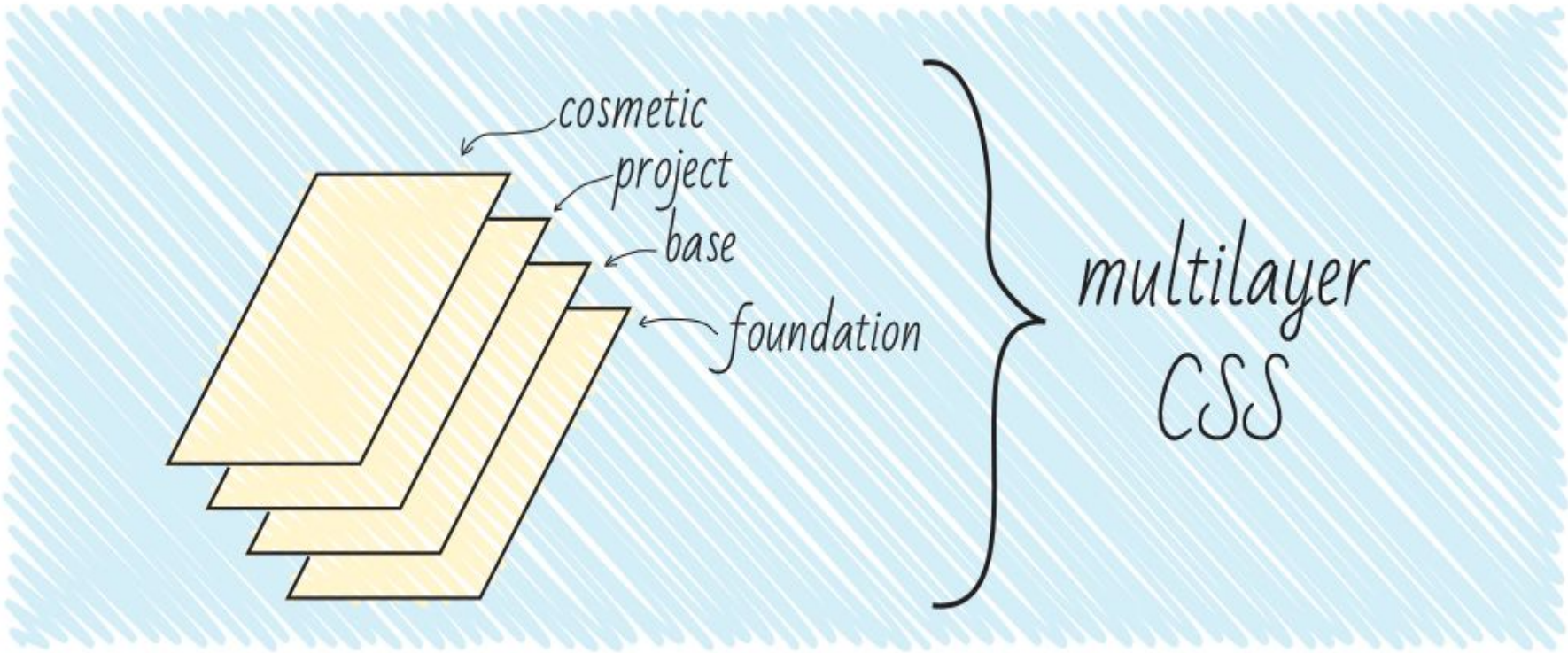
Atomic CSS, редко также ACSS — атомарный CSS. В некотором роде этот подход представляет собой OOCSS, возведенный в абсолют.

При использовании такого подхода для каждого повторно используемого свойства должен быть сформирован отдельный класс. Пример: стиль «margin-top: 1px» предполагает создание класса «mt-1», стиль «width: 200px» создание класса «w-200».

Atomic CSS +/-

- минимизирует объем CSS-кода за счет повторного использования деклараций
- сравнительно легко вводить изменения в модули
- наименования классов представляют собой описательные названия свойств, не описывая семантическую сущность элемента, что иногда может усложнить разработку;
- настройки отображения элементов переносятся непосредственно в HTML

MCSS



MCSS - понятие

MCSS — многослойный CSS (Multilayer CSS). Этот стиль написания кода, предлагает разделить стили на несколько частей, называемых **слоями**.

MCSS - структура

Нулевой слой или **фундамент** — здесь содержится код, отвечающий за сброс стилей браузера (reset.css/normalize.css);

Базовый слой — включает в себя стили многократно используемых на сайте элементов: кнопок, полей ввода для текста, подсказок и т.п.;

MCSS - структура

Проектный слой — включает в себя отдельные модули, а также т.н. «контекст» — модификации элементов в зависимости от браузера клиента, устройства, на котором просматривается сайт/приложение, роли пользователя и т. п.;

MCSS - структура

Косметический слой — в этом разделе находится код, написанный в стиле OOCSS, осуществляющий мелкие изменения в внешнем виде элементов. Здесь рекомендуется оставлять только стили, влияющие на внешний вид и не способные поломать верстку сайта — цвета, некоторые некритичные отступы.

Иерархия взаимодействия слоев

- Слой фундамента задает нейтральные стили и не влияет на другие слои.
- Элементы базового слоя могут влиять только на классы своего же слоя. *Пример: иконки на сайте могут иметь размеры 25x25, но иконки в кнопках — 16x16.*
- Элементы проектного слоя могут влиять на базовый и проектный слой. *Примеры: иконки в форме логина имеют особый размер 20x20; модуль «Покупка» может включать в себя форму логина, стили которой несколько модифицированы.*
- Косметический слой оформлен в виде описательных OOCSS-классов («атомарные» классы) и не влияет на другой CSS-код, избирательно применяясь в разметке.

AMCSS

```
<div am-size="large" am-disabled></div>
```

AMCSS - понятие

Название подхода означает «Модули атрибутов для CSS» (Attribute Modules for CSS). Этот способ является несколько более человеко-читаемым представлением БЭМ-структуры.

Пример

```
<div class="button button--large
button--blue">Кнопка</div>
```

```
<div button="large blue">Кнопка</div>
```

```
<div am-button="large blue">Кнопка</div>
```

Запись кода

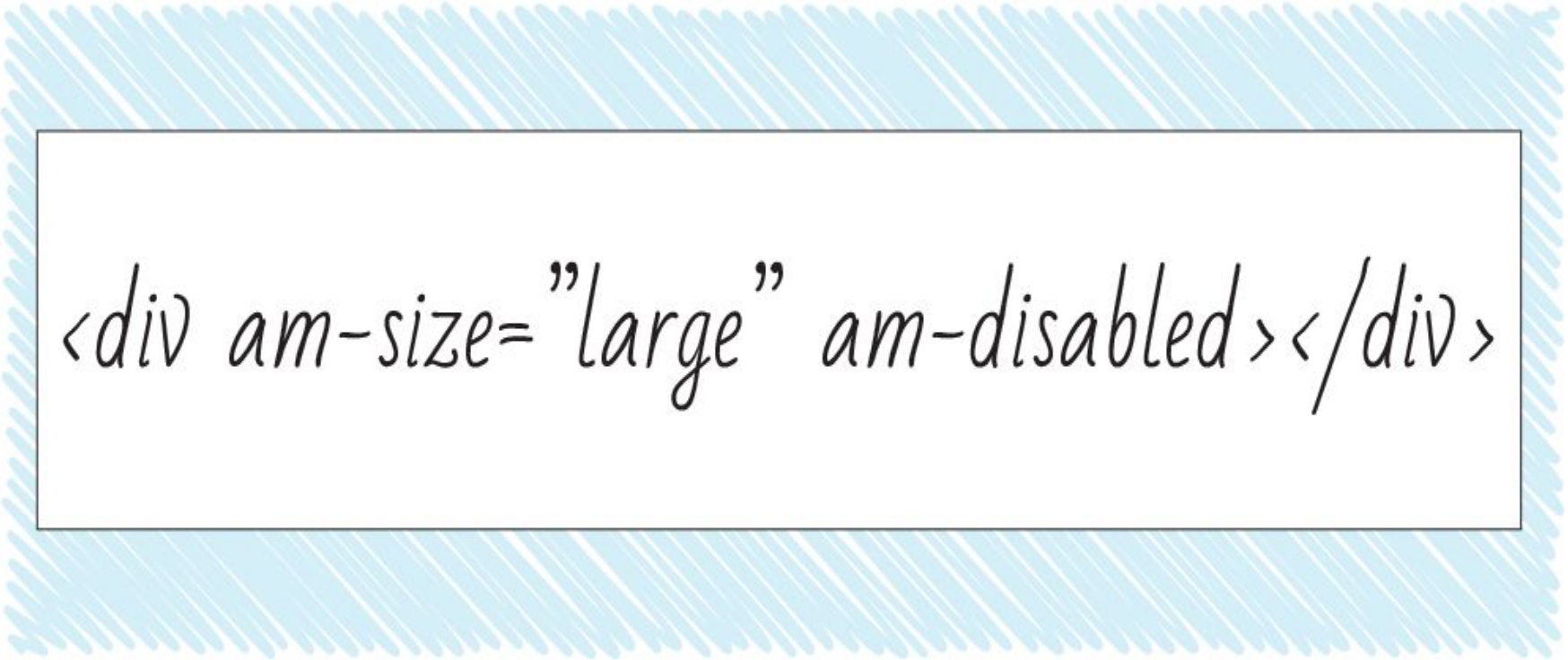
Для записи CSS-кода используется селектор «~=», который работает как атрибут класса: выбирает элементы, значения атрибутов которых содержат указанные слова, разделенные пробелами.

Пример

```
.button {...}
.button--large {...}
.button--blue{...}
```

```
[am-button] {...}
[am-button~="large"] {...}
[am-button~="blue"] {...}
```

FUN



FUN - понятие

FUN означает «плоская иерархия селекторов, служебные стили, компоненты с неймспейсами» (Flat hierarchy of selectors, Utility styles, Name-spaced components).

FUN - понятие

- **F**, плоская иерархия селекторов: в стилях рекомендуется использовать классы для выбора элементов, не вкладывать селекторы, а также не использовать id;
- **U**, служебные классы: поощряется создание служебных атомарных стилей для решения типовых задач верстки;
- **N**, компоненты с неймспейсами: рекомендуется добавлять неймспейсы для задания стилей элементов конкретных модулей; такой подход позволит избежать совпадений в названиях классов.