
# HTML и CSS. Уровень 3.

Продвинутые методологии и инструменты верстки

# О преподавателе

Сурначева Александра Дмитриевна
asurnacheva@specialist.ru

- 2014 - разработка АИСКУЭ для ПАО "МОЭСК"
- 2015 - разработка клиентского офиса для ПАО "МОЭСК"
- 2016 - разработка CMS на Clojure
- 2016 - разработка web-интерфейса для резидентов Сколково
- 2017 - разработка клиентского офиса для ПАО "РусГидро"
- 2017 - front-end разработчик ГУП "МосгортрансНИИпроект"

# Программа курса

http://www.specialist.ru/course/ashtml3
Количество ак.часов: 24

- Методологии верстки
- Препроцессоры
- Компонентная модель верстки
- Макетирование страниц с применением модели гибких блоков, css-grids
- CSS-фреймворки
- 3D и современные методы создания сайтов


# Модуль 1. Методологии верстки

# Если бы мы писали код без методологии...

- Сложности в масштабировании
- Большое количество кода
- Трудности в поддержке кода
- Непонятный код

# Методология Яндекс.БЭМ

# БЭМ (Блок, Элемент, Модификатор) — компонентный подход к веб-разработке

В основе лежит принцип разделения интерфейса на независимые блоки.

Позволяет легко и быстро разрабатывать интерфейсы любой сложности и повторно использовать существующий код.

# БЭМ-сущности

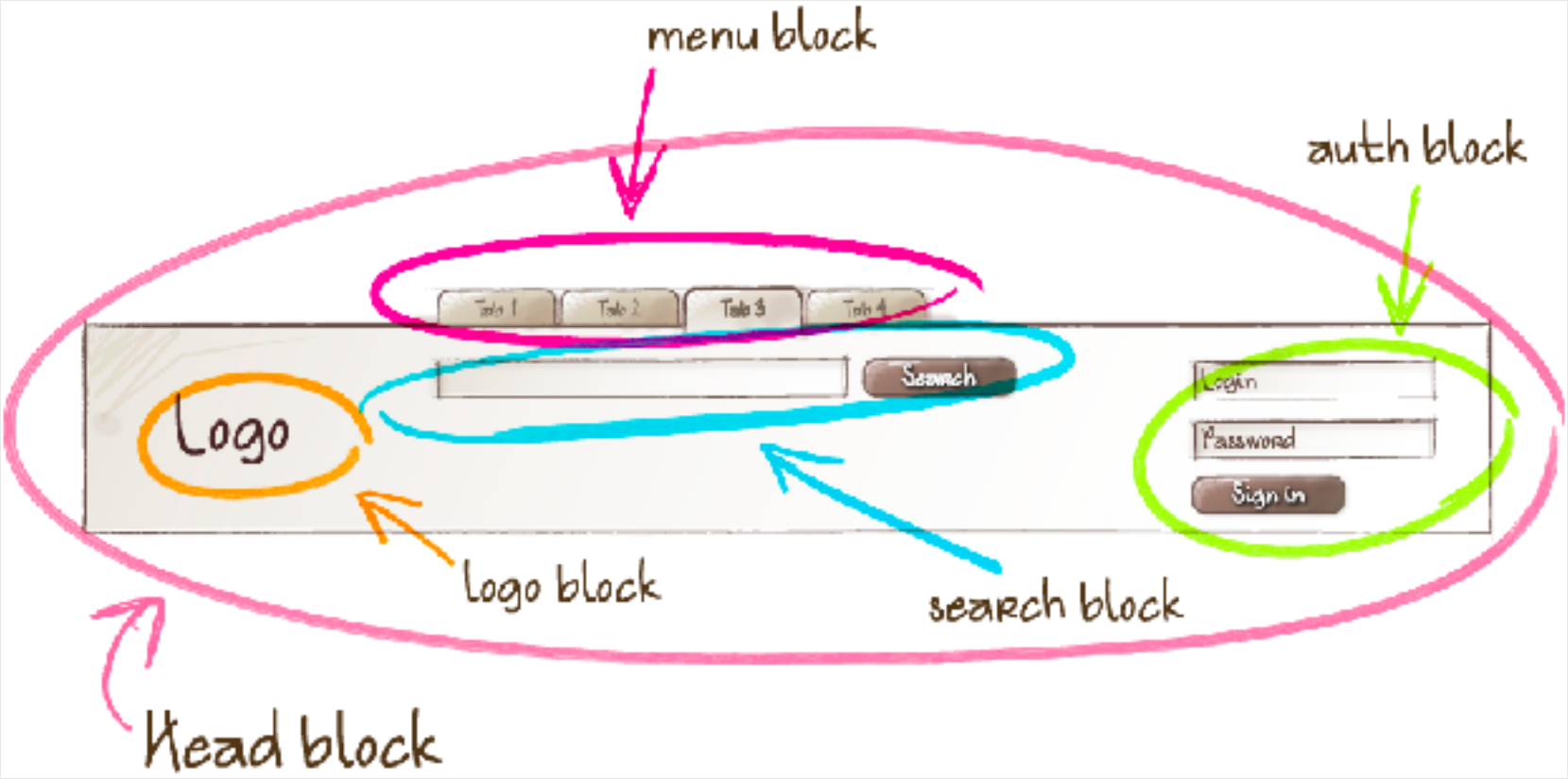
- Блоки
- Элементы
- Модификаторы

# Блок

Логически и функционально независимый компонент страницы, аналог компонента в Web Components.

Блок инкапсулирует в себе поведение (JavaScript), шаблоны, стили (CSS) и другие технологии реализации.

Независимость блоков обеспечивает возможность их повторного использования, а также удобство в разработке и поддержке проекта.

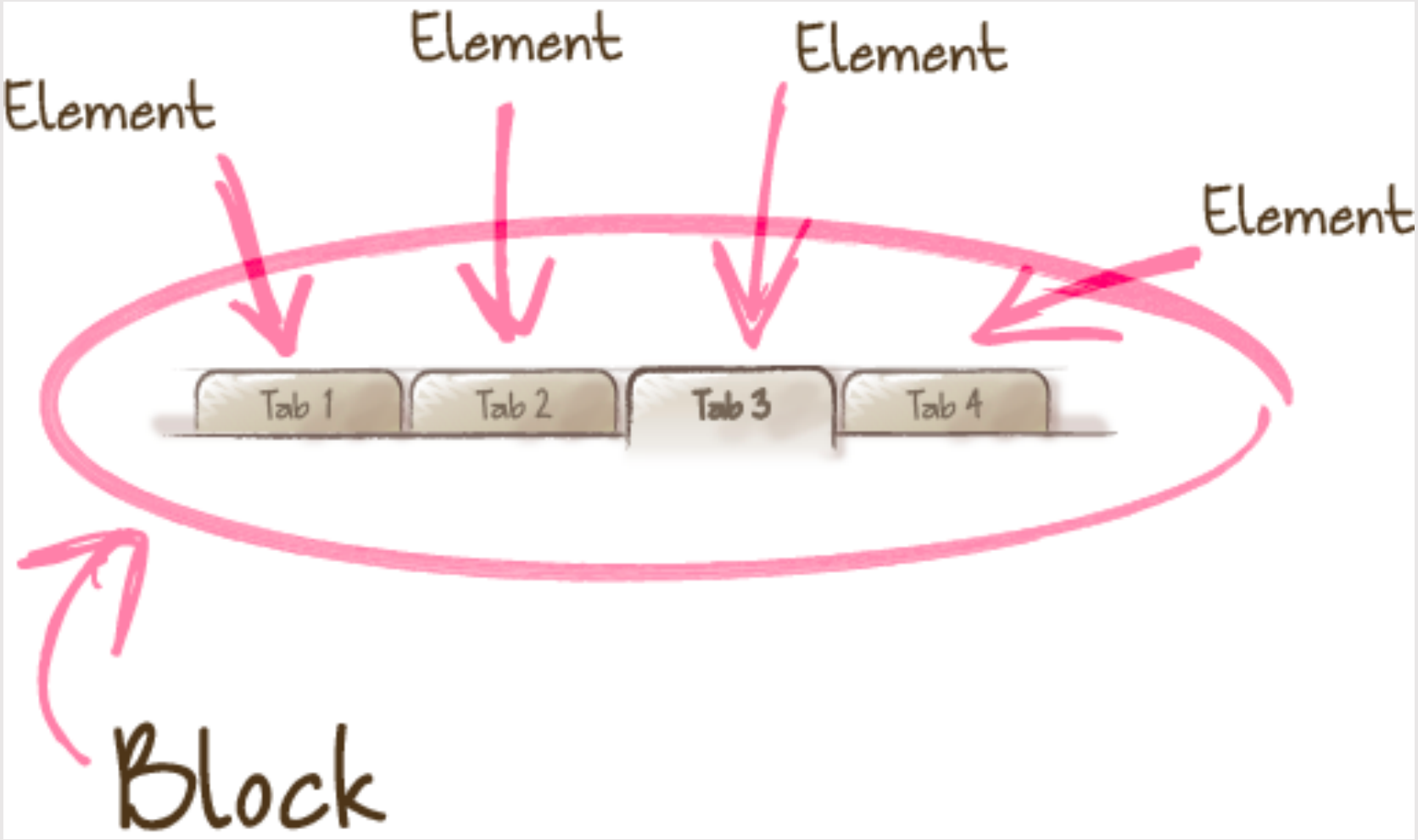

menu block
auth block
Tab 1
Tab 2
Tab 3
Tab 4
Search
Logo
Login
Password
Sign in
logo block
search block
Head block

# Возможности блоков

- Вложенная структура
- Свободное перемещение
- Повторное использование

# Элемент

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


Element
Element
Element
Element
Tab 1
Tab 2
Tab 3
Tab 4
Block

# Модификатор

БЭМ-сущность, определяющая внешний вид, состояние и поведение блока или элемента.

Использование модификаторов опционально.

По своей сути модификаторы похожи на атрибуты в HTML. Один и тот же блок выглядит по-разному благодаря применению модификатора.

# Соглашение по именованию

Основная идея соглашения по именованию — сделать имена CSS-селекторов максимально информативными и понятными. Это поможет упростить разработку и отладку кода, а также решить некоторые проблемы веб-разработчиков.

# menuitemvisible

**menu-item-visible**

**menuItemVisible**

menu — блок
item — элемент
visible — модификатор

# Соглашение по именованию CSS-селекторов

Имена БЭМ-сущностей записываются с помощью латинских букв или цифр в нижнем регистре.

Для разделения слов в именах используется дефис (-).

Для хранения информации об именах блоков, элементов и модификаторов используются CSS-классы.

# Имя Блока

Имя блока создается по схеме:
`block-name`

Имя блока задает пространство имен для элементов и модификаторов.

# Имя Элемента

Пространство имен, заданное именем блока, определяет принадлежность элемента к данному блоку.

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

Полное имя элемента создается по схеме:
`block-name__elem-name`

Если блок имеет несколько одинаковых элементов, как в случае пунктов меню, то все они будут иметь одинаковые имена `menu__item`

Важно! В методологии БЭМ не существует элементов элементов.

# Имя Модификатора

Пространство имен, заданное именем блока, определяет принадлежность модификатора к данному блоку или его элементу.

Имя модификатора отделяется от имени блока или элемента одним подчеркиванием (_).

Полное имя модификатора создается по схеме:

- для булевых модификаторов

```
owner-name_mod-name
```

- для модификаторов вида «ключ-значение»

```
owner-name_mod-name_mod-val
```

Важно! В методологии БЭМ модификатор не может использоваться в отрыве от своего владельца.

# Пример использования БЭМ

Реализация формы аутентификации в HTML и CSS

HTML

```
<form class="form form_login form_theme_forest»>
    <input class=«form__input">
    <input class="form__submit form__submit_disabled»>
</form>
```

CSS

```
.form {}
.form_theme_forest {}
.form_login {}
.form__input {}
.form__submit {}
.form__submit_disabled {}
```

CSS без БЭМ

```
.form {}
.theme {}
.login {}
.input {}
.submit {}
.disabled {}
.forest {}
```

CSS с использованием БЭМ

```
.form {}
.form_theme_forest {}
.form_login {}
.form__input {}
.form__submit {}
.form__submit_disabled {}
```

# Плюсы БЭМ

- Достаточно одного класса
- Специфичность CSS-правил
- Абсолютно Независимые Блоки

# Минусы БЭМ

- Длинные названия классов
- Сложность принятия БЭМ командой разработчиков

# БЭМ не использует селекторы...

- ... по id;
- ... по тегу (элементу);
- ... универсальный селектор;
- ... reset.css/normalize.css (общий сброс стилей)
- ... комбинированные
- ... атрибутов.

https://ru.bem.info

# OOCSS

# Object-Oriented CSS
(Объектно-ориентированный CSS)

**Основная идея -**
многократность использования написанного кода

# Принципы

- Объект - повторяющиеся стили
- Отделение структуры от оформления
- Отделение контейнеров от содержимого

# Стили без OOCSS

```
#button {
        width: 200px;
        height: 50px;
        padding: 10px;
        border: solid 1px #ccc;
        background: linear-gradient(#ccc, #222);
        box-shadow: rgba(0, 0, 0, .5) 2px 2px 5px;
}

#box {
        width: 400px;
        overflow: hidden;
        border: solid 1px #ccc;
        background: linear-gradient(#ccc, #222);
        box-shadow: rgba(0, 0, 0, .5) 2px 2px 5px;
}

#widget {
        width: 500px;
        min-height: 200px;
        overflow: auto;
        border: solid 1px #ccc;
        background: linear-gradient(#ccc, #222);
        box-shadow: rgba(0, 0, 0, .5) 2px 2px 5px;
}
```

# Стили с применением OOCSS

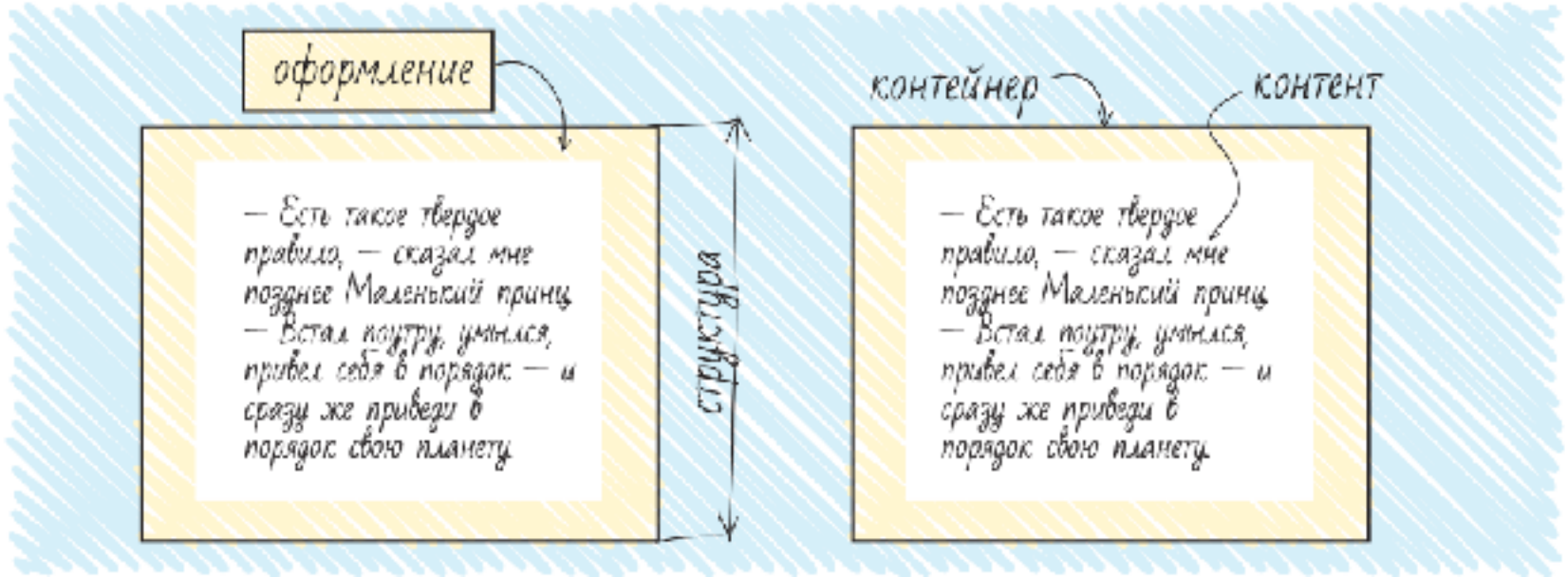
Все стили уникальны:

```
.button {
        width: 200px;
        height: 50px;
}

.box {
        width: 400px;
        overflow: hidden;
}

.widget {
        width: 500px;
        min-height: 200px;
        overflow: auto;
}

.skin {
        border: solid 1px #ccc;
        background: linear-gradient(#ccc, #222);
        box-shadow: rgba(0, 0, 0, .5) 2px 2px 5px;
}
```


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.

# Плюсы OOCSS

- Нет определенных правил
- Более быстрые страницы
- Легко обслуживаемые таблицы стилей

# Минусы OOCSS

- Нет определенных правил
- Задается очень много классов
- Отслеживание существующих модулей

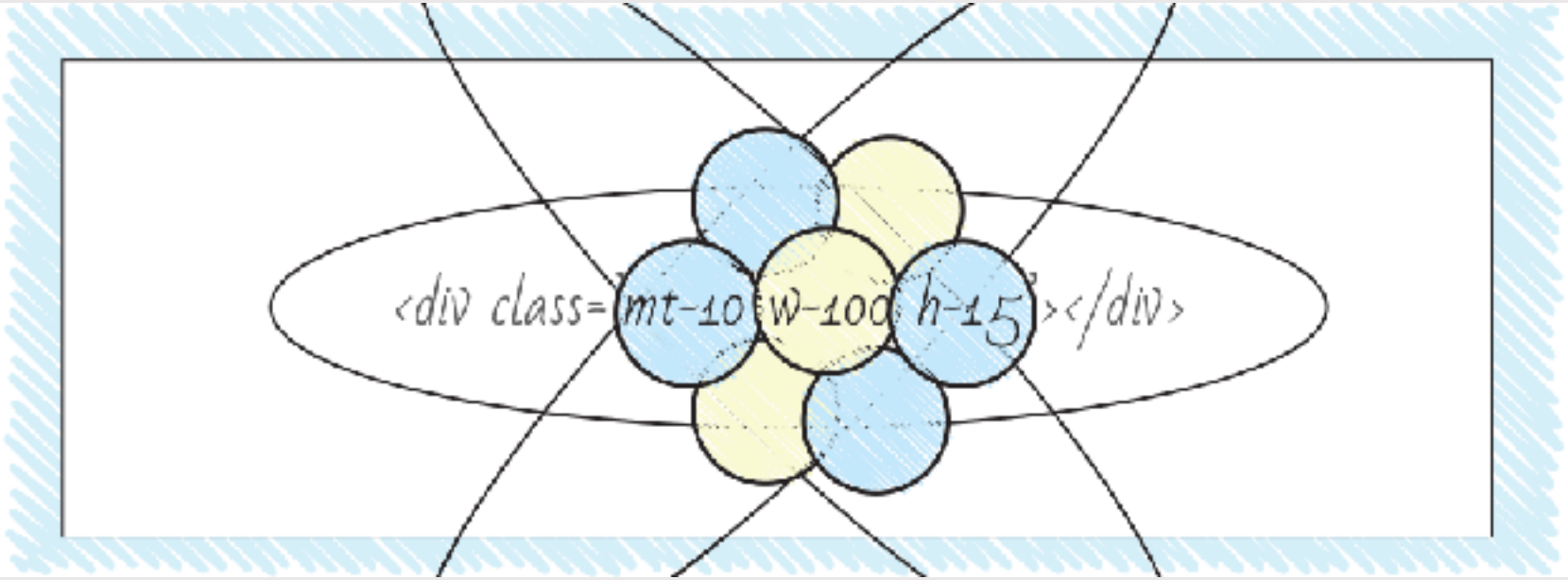
http://oocss.org

# Atomic CSS
(Атомарный CSS)

**Основная идея -**
повторное использование стилей

# Принципы

- Разделение стилей
- Для каждого повторно используемого свойства должен быть сформирован отдельный класс
- Один класс - одно свойство



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».

Такой стиль позволяет минимизировать объем CSS-кода за счет повторного использования деклараций, а также сравнительно легко вводить изменения в модули, к примеру, при изменении технического задания.

# Плюсы Atomic CSS

- Небольшой объем CSS
- Легко вводить изменения
- Возможность повторного использования

# Минусы Atomic CSS

- Непонятные названия классов
- Настройки отображения элементов переносятся непосредственно в HTML

# SMACSS

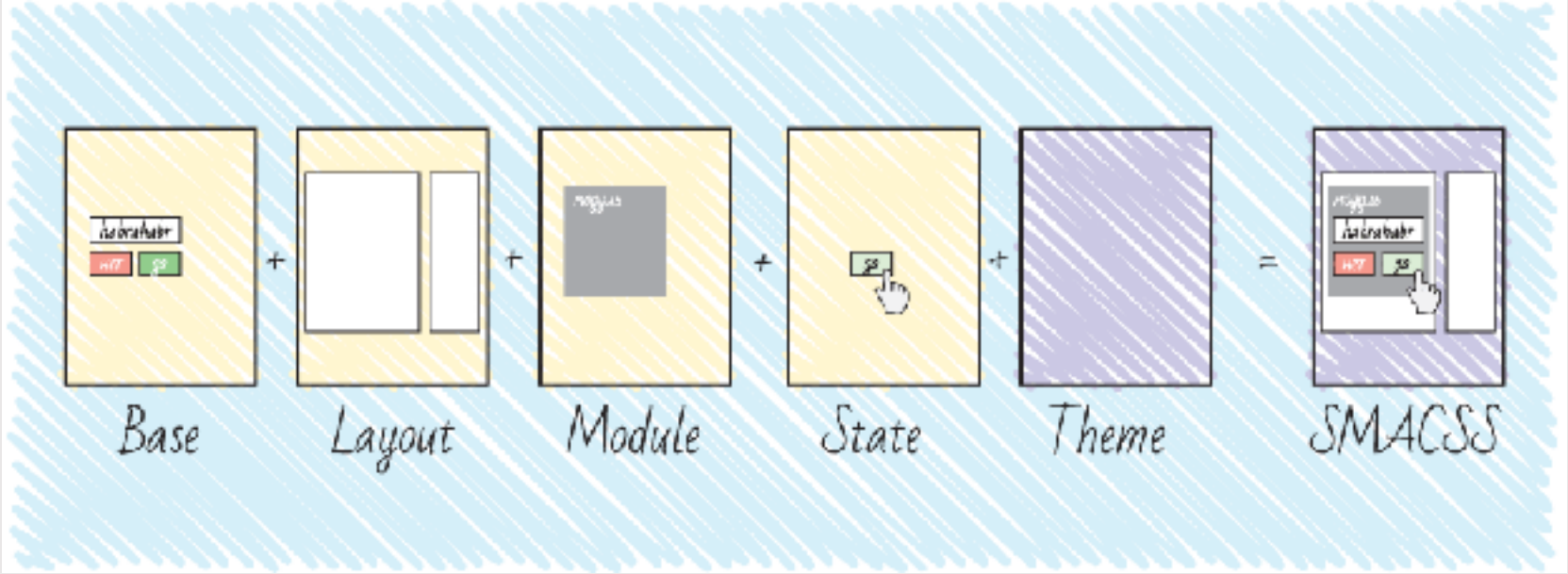
# Scalable and Modular Architecture for CSS (Масштабируемая и модульная архитектура для CSS)

**Основная цель -**
уменьшение количества кода и
упрощение поддержки кода

**Основная идея -**
разделение стилей на 5 составляющих

# Составляющие SMACSS

- Base rules (базовые стили)
- Layout rules (стили макета)
- Modules rules (стили модулей)
- State rules (стили состояний)
- Theme rules (стили оформления)


Base
Layout
Module
State
Theme
SMACSS

# Base rules

Базовые стили. Это стили основных элементов сайта — body, button, a и т.д.

В этой секции используются в основном селекторы элементов и атрибутов, классы — в исключительных случаях (например, если у вас стилизованные JavaScript'ом селекты)

**Example Base Styles**

```
body, form {
    margin: 0;
    padding: 0;
}

a {
    color: #039;
}

a:hover {
    color: #03F;
}
```

# Layout rules

Стили макета. Здесь находятся стили глобальных элементов - размеры шапки, подвала, сайдбара и т.д.

Layout declarations

```
#header, #article, #footer {
    width: 960px;
    margin: auto;
}

#article {
    border: solid #CCC;
    border-width: 1px 0 0;
}
```

# Modules rules

Стили модулей, то есть блоков, которые могут использоваться несколько раз на одной странице.

**Module example**

```
.module > h2 {
    padding: 5px;
}

.module span {
    padding: 5px;
}
```

# State rules

Стили состояния. В этом разделе прописываются различные состояния модулей и скелета сайта.

Это единственный раздел, в котором допустимо использование ключевого слова «!important».

State applied to an element

```
<div id="header" class="is-collapsed">
    <form>
        <div class="msg is-error">
            There is an error!
        </div>
        <label for="searchbox" class="is-hidden">Search</label>
        <input type="search" id="searchbox">
    </form>
</div>
```

# Theme rules

Стили оформления. 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.).

**Module Theming**

```
// in module-name.css
.mod {
    border: 1px solid;
}

// in theme.css
.mod {
    border-color: blue;
}
```

# Плюсы SMACSS

- Управляемый код
- Расширяемый код
- Возможность повторного использования
- Дополнительные уровни семантики

# Минусы SMACSS

- Непривычно использовать
- Надо много думать и четко следовать всем правилам

https://smacss.com

# Оформление кода
## (правила оформления кода от Google)

https://google.github.io/styleguide/htmlcssguide.html

перевод:
https://habr.com/post/143452/