Длинная телега про Бутстрап

[Raw](https://gist.github.com/iAdramelk/d328b73c72cab92ef95f/raw/a2b848a215d36cbba64fde1b112b6c2a7137cfdb/.md)

[**.md**](https://gist.github.com/iAdramelk/d328b73c72cab92ef95f#file-md)

**Вводная часть**

У CSS есть несколько базовых проблем, которые позволяют очень быстро отстрелить себе ногу при неправильном использовании:

1. Глобальный неймспейс – в серверном программировании все что написано в файле, в файле и остается. Все же что написано в css и js засирает глобальное пространство имен со всеми вытекающими. В JS эту проблему сейчас побороли всякими модульными системами, а вот с css сложнее. В идеальном мире это должен починить Shadow DOM и настоящие Web Components, но пока их нет единственный способ с этим бороться – следовать какой-то системе именований селекторов, которая по возможности уменьшает и исключает возможные конфликты.
2. Каскадность – если на один элемент может сработать несколько правил, то они все и сработают последовательно. Если есть элемент h1.title, на него сработают все правила для тегов h1 и все правила для класса .title. Так как весь html состоит из тегов, то правил которые применяются на теги без классов будут работать на все вообще.

Соответственно назначать или переназначать стили у тегов – это примерно то же самое, что править прототипы объектов в JS, чем в свое время печально славилсяPrototype.js. Эти стили унаследует вообще все объекты и если их потом захочется поменять, то результат будет такой же, как если ты решил в прототипе объекта поменять результаты какого-то метода, который используют все дети этого объекта. Вероятность что-то сломать почти 100%.

1. Вложенные селекторы. Можно написать селекторы .nav .item {...} и .menu .item и .item в зависимости от места вывода будет показываться по-разному. Все хорошо пока тебе не нужно поместить блок menu внутрь блока nav. Тогда сайдэффекты становятся совершенно непредсказуемые. По сути аналог вложенных селекторов из программирования – это функция которая в зависимости от места где её вызывают, выдает разный результат. Например в одном месте sum(2,2) может возвращать 3, а в другом 5.

**Зачем нужны методологии**

Хорошая методология занимает предотвращением этих проблем. Покажу как это делает БЭМ, но CSS Modules, Polymer или всякие решения с инлайновыми стилями для Реакта тоже решают именно их, только другим способом.

Как именование классов по БЭМу помогает решать эти проблемы:

1. БЭМ запрещает применять стили на теги, максимум ресет. На id тоже нельзя, потому что такие элементы нельзя на странице использовать 2 раза, а сколько раз он тебе понадобится ты не всегда знаешь заранее. Все стили можно применять только к классам.
2. БЭМ создает для всех компонентов глобальный неймспейс – все классы которые относятся к компоненту начинаются с одного префикса. Это позволяет исправить второй пример таким образом: .nav\_\_item, .menu\_\_item. Если один вложить в другой не будет конфликта правил.
3. Под каждый компонент в БЭМ создается своя папка – это защищает тебя от конфликтов в именах компонентов и при правильном использовании дает гарантию, что в глобальном неймспейсе будет только один компонент с таким префиксом.
4. В БЭМ есть только один вид вложенных селекторов: модификатор > элемент. Оба начинаются с одного префикса, оба живут в одном файле, оба никак не влияют на другие компоненты.

**Что делает Bootstrap**

Bootstrap нарушает КАЖДОЕ из этих правил:

1. Bootstrap переназначает стили тэгов.
2. Bootstrap в куче случаев меняет способ отображения элемента в зависимости от того, кто его родители. Хорошо хоть сейчас делает это через >, а не просто так. Но вот<https://github.com/twbs/bootstrap/blob/master/less/button-groups.less> такие штуки все равно сильно уменьшают предсказуемость и усложняют редизайн.
3. Bootstrap загрязняет глобальный неймспейс сотнями классов с очень generic именами: .table, .dropdown, .row, .left, и т. д. Которые надо все помнить и ни в коем случае не использовать самому.

При таком подходе отстреливание себе ноги становится только вопросом времени.

**Когда Boostrap можно использовать**

Чтобы не отстрелить себе ногу Bootstrap'ом и получить от него пользу нужно чтобы проект соответствовал ряду требований:

1. На проекте много страниц.
2. Страницы собираются из простых базовых элементов. Базовый – это кнопка или таблица. Сложный – это пост в фейсбуке или комментарий к нему, они состоят из нескольких тегов и их нужно переносить и реиспользовать все вместе.
3. Нет мест, где изменение и эксперименты с дизайном могут сильно повышать/понижать конверсию или другие важные метрики. Как корзина или страница товара в интернет магазине.
4. Никогда не будет глобального редизайна.
5. Типизация страниц окупается скоростью их внедрения.

Еще можно для прототипов и быстрого старта, чтобы потом выкинуть. Для остальных случаев это боль скорее.

**Примеры ситуаций когда Bootstrap НЕ подходит**

**1. Большие интернет-магазины и новостные сайты**

У интернет-магазина есть главный KPI – конверсия в покупки. Покупки очень сильно зависят от дизайна страницы товара и корзины/процесса оформления заказа. Изменение конверсии во многом зависят от дизайна, прибавка на полпроцента может тебе позволить удвоить размер команды фронтендеров.

Тут ты дизайнера не запинаешь в гайды. Наоборот ему нужно давать максимум возможностей для экспериментов и изменений, брать эти варианты и гонять 50 сплит-тестов одновременно. С разными отступами, размерами шрифтов, кастомными элементами и прочим.

Если делать такое на Бутстрапе без использования какой-нибудь методологии, то очень просто незаметно сломать что-то в какой-то невидимой сейчас части сайта или в каком-то редком состоянии текущей страницы. Ну и процесс подгонки по большей части будет состоять из борьбы с Бутстрапом и его локальными переназначениями.

Примеры:

<http://www.amazon.com/> – тысячи страниц, сотни сплиттестов, постоянный ползучий редизайн. <http://www.gog.com/> – мало стандартных элементов и много кастомных под этот сайт.<http://www.nytimes.com/> – очень много сложных компонентов и исключений. Несколько ключевых экранов от дизайна которых зависят KPI. Сюда же большинство сайтов больших СМИ. <http://arzamas.academy/> – так, до кучи.

**2. Сайты со сложными реиспользуемыми в разных местах компонентами**

Довольно часто компоненты которые тебе надо реиспользовать – это не кнопки и формы, а что-нибудь сильно больше по размеру. Например пост на Фейсбуке или лента с комментариями оттуда же. Бутстрап тебе никак не поможет тебе выделить этот компонент и защитить его от сайдэффектов. Тебе все равно придется использовать какую-нибудь методологию.

А вот вызвать проблемы, когда ты вставляешь этот элемент в какое-то новое место можно довольно легко.

Самый простой вариант взять сложный компонент в котором используется .form-control или label и поместить его в какую-нибудь форму с модификаторами. Куча разнообразный сайдэффектов обеспечена: <https://github.com/twbs/bootstrap/blob/master/less/forms.less>

Также много интересного можно получить если ты хочешь сделать выпадающий компонент (как непрочитанные сообщения в фейсбуке) и разместить его в закрытом состоянии в шапке, рядом с кнопкой которая его открывает: <https://github.com/twbs/bootstrap/blob/master/less/navbar.less>

Примеры:

<https://mail.yandex.ru/> – мало стандартных контролов, куча кастумных контролов, много исключений, поддержка легаси. <http://trello.com/> – много сложных компонентов, не очень много стандартных бутстраповских. Для <http://try.discourse.org/> польза тоже весьма относительная будет.

**3. Сайты с большим количеством типов страниц, которым может потребоваться редизайн**

Сюда хорошо подходят те же интернет-магазины, большие онлайновые журналы/газеты, городские порталы.

Допустим ты решил поменять кегль у заголовков на каком-нибудь [http://www.e1.ru](http://www.e1.ru/) . Ты переназначил стили h1 и h2. Ну ой, теперь тебе надо протыкаться по 100+ типам страниц и убедиться что у тебя не сломалось ничего: Ничего не вылазит за границы блоков.

При этом помня что еще <https://github.com/twbs/bootstrap/blob/master/less/jumbotron.less> и другие переназначатели и что изменения базовых стилей их скорее всего поломает.

Постепенно (постранично или даже по частям страниц) переводить сайт на новый дизайн при этом продолжая использовать стандартные бутстраповские элементы не получится.

Примеры:

<https://www.kickstarter.com/> – за последние пару лет сайт как минимум дважды глобально передизайнивали, не считаю отдельных локальных экспериментов.

**4. Промосайты и лэндинги**

Из всего Бутстрапа нам нужны заголовки, две кнопки и два поля формы. Остальное параллаксы, видео, большие иконки и прочее. Сайт нужен на месяца, после этого его выключат. Борьба с бутстрапом тут съест больше времени, чем пользы принесет.

Как пример: <http://um.mos.ru/promo/gosudarstvennyy_istoricheskiy_muzey/>

**5. Игры в браузерах**

No comments.

**Резюме**

Да, у Бутстрапа есть своя зона применения. Но она точно не 80% сайтов. К 20-30% он подходит хорошо. Еще на столько же его можно натянуть, с весьма вероятными потенциальными проблемами. В остальных случаях сразу нет.

ну и надо всегда помнить, что проект обычно начинается как что-то простое и предсказуемое, а потом в процессе мутирует во что-то совсем другое. В случае с Bootstrap это будет большой проблемой, в случае с BEM или CSS Modules оно будет просто работать.

[](https://gist.github.com/vayser)

[**vayser**](https://gist.github.com/vayser)commented [on 4 Oct](https://gist.github.com/iAdramelk/d328b73c72cab92ef95f#gistcomment-1588134)

20% явно заниженная оценка. Бутстрап отлично настраивается если внедрить его сборку в свой проект. Главное выработать подход, понять что к чему в variables.less и 20% превратиться в 80%. Прирост скорости всегда оправдывает потраченное на framework время в моих случаях

[](https://gist.github.com/meuwka)

[**meuwka**](https://gist.github.com/meuwka)commented [on 4 Oct](https://gist.github.com/iAdramelk/d328b73c72cab92ef95f#gistcomment-1588216)

[@vayser](https://github.com/vayser) а причем тут настройка. настроить легко, хотя я сталкивалась с ситуациями, что не всё нужное вынесено в переменные, куча лишней стилизации, которую потом приходится перекрывать.

прирост скорости только в начале, если проект развивается и чаще всего дизайн начинает выходить за рамки начального, начнется конкретный упадок скорости, по сравнению с изолированными блоками.

за пару проектов по методологии можно выработать уже свой собственный бутстрап, который потом из проекта в проект таскать можно.

[](https://gist.github.com/dshster)

[**dshster**](https://gist.github.com/dshster)commented [on 4 Oct](https://gist.github.com/iAdramelk/d328b73c72cab92ef95f#gistcomment-1588287)

Насколько я помню Bootstrap разрабатывался для быстрого прототипирования интерфейсов, когда бек-разработчик без умения верстать может быстро накидать пару форм и менюшек в админке под свой функционал.  
Кто придумал делать на Bootstrap чистовой дизайн - непонятно. Со всеми его проблемами я столкнулся сразу на первом проекте. И да, я тщательно изучил variables.less, подогнал всё что можно под дизайн, а потом плевался в попытке редизайнить то, что предлагал Bootstrap и в итоге стал использовать только сетку.

[](https://gist.github.com/CreativeSeo33)

[**CreativeSeo33**](https://gist.github.com/CreativeSeo33)commented [on 5 Oct](https://gist.github.com/iAdramelk/d328b73c72cab92ef95f#gistcomment-1588360)

Использую в основном только сетку и миксины Sass. Стили кнопок, таблиц, форм, и т.д. не вижу смысла использовать, только если дизайна админок.

[](https://gist.github.com/elky)

[**elky**](https://gist.github.com/elky)commented [on 5 Oct](https://gist.github.com/iAdramelk/d328b73c72cab92ef95f#gistcomment-1588671)

[@dshster](https://github.com/dshster)

Кто придумал делать на Bootstrap чистовой дизайн - непонятно.

Вероятно заказчики, которым кто-то промыл мозг относительно того, что это быстро и мобайл рэди. Сам к бутстрапу всегда относился скептически, но когда на фрилансерских ресурсах типа Upwork заказчик с хорошим бюджетом в качестве обязательного условия пишет "Bootstrap only", и переубедить его не получается, то приходится изворачиваться с бутстрапом.

[](https://gist.github.com/saintbyte)

[**saintbyte**](https://gist.github.com/saintbyte)commented [on 5 Oct](https://gist.github.com/iAdramelk/d328b73c72cab92ef95f#gistcomment-1588807)

Теперь есть на кого ссылать почему я использую классы , а вообще ждем победы веб-компонентс

[](https://gist.github.com/e-asphyx)

[**e-asphyx**](https://gist.github.com/e-asphyx)commented [on 5 Oct](https://gist.github.com/iAdramelk/d328b73c72cab92ef95f#gistcomment-1588847)

В JS эту проблему сейчас побороли всякими модульными системами

А можно поподробнее?

[](https://gist.github.com/kof)

[**kof**](https://gist.github.com/kof)commented [on 5 Oct](https://gist.github.com/iAdramelk/d328b73c72cab92ef95f#gistcomment-1589126)

[я вот так вот решаю все эти проблемы](https://github.com/jsstyles/jss)

[](https://gist.github.com/iAdramelk)

Owner

[**iAdramelk**](https://gist.github.com/iAdramelk)commented [on 6 Oct](https://gist.github.com/iAdramelk/d328b73c72cab92ef95f#gistcomment-1589236)

[@e-asphyx](https://github.com/e-asphyx) Я имею ввиду [CommonJS](http://requirejs.org/docs/whyamd.html#commonjs) и [AMD](http://requirejs.org/docs/whyamd.html#amd) модули. Если писать JavaScript в таком формате и использовать инструменты вроде Require.js, Browserify или Webpack, то можно не волноваться про глобальные переменные. Плюс в обозримом будущем браузеры из коробки будут поддерживать [ES6 Modules](http://eviltrout.com/2014/05/03/getting-started-with-es6.html).

[](https://gist.github.com/artin-phares)

[**artin-phares**](https://gist.github.com/artin-phares)commented [on 7 Oct](https://gist.github.com/iAdramelk/d328b73c72cab92ef95f#gistcomment-1590857)

в Реакте кстати можно решать не только инлайновыми стилями - [react-jss](https://github.com/jsstyles/react-jss)  
генерится уникальный css класс для компонента, и потому нет проблем с псевдоклассами/@-правилами, и ворохом дублированного инлайна на Elements tab.  
и в тоже время цель достигнута: нет каскада и переопределений - стили аккуратно изолированны под компоненты.

[](https://gist.github.com/GarikFF)

[**GarikFF**](https://gist.github.com/GarikFF)commented [on 8 Oct](https://gist.github.com/iAdramelk/d328b73c72cab92ef95f#gistcomment-1591568)

хорошая статья.   
мне кажется очень многие со временем приходят к концепции Абсолютно Независимых Блоков естественным путем. Просто потому что это безумно удобно и чертовски логично.

[](https://gist.github.com/murashki)

[**murashki**](https://gist.github.com/murashki)commented [on 8 Oct](https://gist.github.com/iAdramelk/d328b73c72cab92ef95f#gistcomment-1591631)

Ты переназначил стили h1 и h2. Ну ой, теперь тебе надо протыкаться по 100+ типам страниц и убедиться что у тебя не сломалось ничего

Странное утверждение. Переопределяем стили у h1 и h2 и далее сваливаем все на бутстрап, что дескать у нас все из-за него везде поломалось )) Бутстрап стилизует эти элементы и прекрасно. Мы полагаем, что все **голые** заголовки в нашем проекте (т.е. которые без классов) будут выглядеть именно так. Определились. Отлично. Меняем стили заголовков - не проблема. Но нужно понимать, что они изменяться везде в проекте, где используются заголовки без стилей. Более того, это и должно произойти и это хорошо, что произойдет, это просто чудесно. В этом вообще-то первостепенная задача стилей, описывать их в одном месте, а применять - во многих.

Хотим для какой-то конкретной страницы поменять стиль заголовков? Почему бы и нет? Сделаем отдельный стиль какой нужно и воткнем на страницу заголовок с этим стилем.

Просто не могу понять какой другой вариант? Не переопределять браузерные стили, а например создать стиль специально для заголовков и юзать его везде? Отлично. Создали стильdefaultPageHeader. Начали верстать страницы. Сверстали 100500 страниц. Пришел один специалист и сказал: "*В этом проекте все так сложно, все так плохо. Вот я переназначил стили для класса defaultPageHeader. Ну ой, теперь мне надо протыкаться по 100+ типам страниц и убедиться что у меня не сломалось ничего*"...

Занавес...

[](https://gist.github.com/murashki)

[**murashki**](https://gist.github.com/murashki)commented [on 8 Oct](https://gist.github.com/iAdramelk/d328b73c72cab92ef95f#gistcomment-1591657)

И потом, у нас был в проекте один такой парень, у которого тоже могло поломаться многое, если где-то в проекте что-то менялось.

Суть такова. Если ты используешь где-то обычный h1, ты говоришь тем самым: "*Меня устроит заголовок такого стиля, который будет определен для проекта*". Если твой компонент чувствителен к line-height - обеспечь это, укажи явно. Если чувствителен к отступам, к размеру текста или его цвету - укажи это. Если твой заголовок таков, что должен умещаться в одну строку, но не в две - обеспечь это. Не нужно ждать, когда размер текста увеличится или со стороны сервера придет длинный заголовок и все поползет.

Далее. h1, h2 и т.д. должны использоваться именно как заголовки. Если в твоем компоненте заголовок чувствителен к размеру текста, к отступам и т.д., тут нужно задуматься, а действительно ли это заголовок и действительно ли нужно использовать именно h1 и h2? Может нам нужно использовать голый div. Переопределять стили для div никто не рискнет - настраивай как угодно. Типичная ошибка верстальщиков использовать p везде невпопад, в то время как его нужно использовать только в тексте, в качестве разделения абзацев. Ясное дело, у таких верстальщиков все поломается, когда мы решим сделать для абзацев отступ больше, чем значение по умолчанию.

Считаю пункт про переопределение стандартных стилей не состоятельным ))

[](https://gist.github.com/iAdramelk)

Owner

[**iAdramelk**](https://gist.github.com/iAdramelk)commented [on 9 Oct](https://gist.github.com/iAdramelk/d328b73c72cab92ef95f#gistcomment-1591961)

[@murashki](https://github.com/murashki) О! Холивар, холивар, я ждал!

Я думаю разработчики [Prototype.js](http://prototypejs.org/) тоже так считали. :) Давайте мы добавим ко всем массивам в JS метод map, что может пойти не так? Дело, напоминаю, было еще до ES5 и синтаксис map в первых версиях Prototype отличался от того что потом в спеке написали. Это начало со временем приводить к интересным эффектам при попытках изменить сайты, где он применялся.

Но это так, лирическое отступление. Теперь по сути.

**Еще раз про переопределение стандартных стилей**

Чтобы понять почему переопределение стандартных стилей достаточно добавить в свой css правило a { display: block; } и посмотреть что получится. :)

Но тут мне конечно скажут, что никто в здравом уме не будет так делать и мы говорим про стили заголовков и текста, так что не надо передергивать.

Давайте поправим что-нибудь более безопасное, например стили списков. Хочу чтобы у них сверху-снизу был padding в 5px. Добавлю-ка я стили для li вот сюда:<https://github.com/twbs/bootstrap/blob/master/less/type.less#L167>

Ой а почему у нас сломались все меню и навигация? Упс, там тоже теги без классов и padding'и они не перебиваются: <https://github.com/twbs/bootstrap/blob/master/less/navs.less#L15>

Но это я, наверное, опять передергиваю, зачем нам править li, мы же говорим про заголовки?

С заголовками все сложнее. Надо сначала разобраться откуда они берутся и зачем используются.

**Зачем они там нужны:**

1. Для визуальной иерархии. Тут все относительно просто.
2. Для CEO – у продвиженцев могу оказаться свои хотелки по поводу того что должно быть написано каким тегом и оно не всегда стыкуется с тем как все на странице выглядит. Например они могут захотеть, чтобы на странице был только один тег h1 или h2, а больше никаких заголовков не было. В Бутстрапе, конечно, есть классы .h1 и .h2, но мы же щас про переопределение тегов говорим. :)
3. Заголовки используют скрин ридеры и это для них основная навигация. Структура, которую мы хотим для них сделать тоже не всегда такая же как для браузера.
4. Заголовки используют читалки вроде Instapaper, от них мы тоже потенциально может захотеть чего-нибудь странного.

**Откуда они берутся:**

1. Их пишет команда.
2. Мы ставим плагины для, например jQuery или компоненты для React и там ВНЕЗАПНО тоже есть заголовки. Незадача.

**Какие жизненные этапы бывают у проекта:**

1. Мы пишем код в первый раз.
2. Мы делаем глобальный редизайн всего сайта из 100500 страниц.
3. Мы сдали проект и клиент передал его на поддержку другой компании, которая не знает всех тонкостей связанных с разными местами где у нас используются заголовки и им пришла задача на редизайн. Ой.

Разделение семантики (тегов) и внешнего вида во всех описанных случаях нам сильно облегчает жизнь. А от тегов может быть куча всего непредсказуемого.

Ну и сразу возникает вопрос правда ли первостепенная задача стилей – давать все переопределять в одном месте, а не в том, чтобы быть уверенным что у пользователя ничего не сломается?

Есть огромное количество задач, когда стабильность важнее единообразия. Возьмем в пример тот же Амазон, количество типов страниц и блоков с заголовками на нем исчисляется тысячами, а если брать сплит тесты, то и десятками тысяч. Если там просто поменять стили на тегах заголовков, это может поломать десятки экранов и уронить конверсию. А падение конверсии для Амазона может стоить десятки миллионов долларов в день запросто.

**По конкретным предложениям**

**Создатель блока должен сделать так, чтобы блок не ломался с любым размером, цветом и формой заголовка, это его обязанность.**

Ээээ. Ну вот допустим у нас есть тег h3 с такими стиялми:

h3 {

font-size: 24px;

color: #000;

}

Он используется в шапке плашки с зеленым фоном, плашка выводится в боковой колонке шириной 220px. Мы решаем сделать редизайн и наш новый заголовок такой:

h3 {

font-size: 48px;

color: green;

padding-left: 50px;

background: ...;

}

Что я, как автор, блока могу сделать чтобы у меня новый зеленый заголовок был виден на зеленом фоне и чтобы текст в два раза большего размера нормально входил в колонку которая стала170px вместо 220px из-за отступа с виньеткой?

Ну и мы внезапно повысили время разработки и поддержки блока на порядок, ведь на все возможные варинты шрифтов, размеров и цветов нам надо сайт протестировать. И это все ради того чтобы оставить возможность стилизовать несчастные h2.

**Для страницы где нужны нестандартные заголовки сделаем отдельный стиль и ставем его там где надо. Проблема решена.**

Ок, вставим Вопрос как вставим:

1. Классом для тега заголовка, например h1.big-title? Чем больше таких ситуаций, тем опаснее переопределять h1. Все комбинации и возможные последствия никто не сможет предсказать.
2. Вложенным селектором, например .news h1? Все ровно то же самое, что и в прошлом пункте.
3. Дивом с классом, например div.big-title? Ну вот мы собственно и пришли к тому, что не надо было со стилями на тегах и начинать возиться.

**Так что делать-то?**

Возможны варианты. Общая идея в том, что заголовки каждого блока нужно уметь менять независимо от всего остального.

Мы у себя этого достигаем следующим образом:

1. Мы не делаем общих классов вроде .h1. Заголовок всегда часть большого блока. Например .review\_\_title или .collection\_\_title.
2. Мы добавляем стандартные стили заголовка миксином в sass или postcss. Он разворачивается в font-size, line-height, letter-spacing, color и все остальное что нам надо.
3. Когда мы хотим поменять стили заголовка, мы создаем новый миксин, обходим вручную каждый блок где он используется и по очереди меняем и тестируем.

Теоретически, у нас остается возможность, изменив миксин, поправить махом стили везде и если у нас страниц мало и мы любим жить рисково мы можем ей воспользоваться. Но  
мы стараемся так не делать все равно.

[](https://gist.github.com/murashki)

[**murashki**](https://gist.github.com/murashki)commented [on 9 Oct](https://gist.github.com/iAdramelk/d328b73c72cab92ef95f#gistcomment-1592221)

Ваша мысль понятна, спасибо за развернутый ответ, весь коммент внимательно прочитал, но я останусь при своем мнении. По многим пунктам не согласен, ваши доводы не возымели эффекта, но как вы правильно подметили, пошел холивар. Поэтому остановимся на этом.

Вам успехов. Пусть ваши проекты будут такими, что бы ничего не ломалось. И пусть тандем из дизайнера, который сначала изобрел зеленые подложки в некоторых блоках, а потом внезапно решил изменить цвет заголовков во всем проекте на тот же зеленый цвет (это гениально, ей Богу); так вот этот дизайнер в купе с программистом, который изначально вставил голый тег h1 на зеленом фоне (в то время как проект верстается на дефолтном, предположим, белом фоне); пусть этот тандем работает слажено и отдает отчет своим деяниям ))

Если все же вы хотите получить ответы на вопросы прозвучавшие в вашем последнем комментарии, скажите какие - с удовольствием прокомментирую ))

[](https://gist.github.com/iAdramelk)

Owner

[**iAdramelk**](https://gist.github.com/iAdramelk)commented [on 10 Oct](https://gist.github.com/iAdramelk/d328b73c72cab92ef95f#gistcomment-1592780)

[@murashki](https://github.com/murashki) ну собственно у меня один вопрос только: Но зачем? :) То есть да, в некоторых ситуациях, когда:

* у сайта утверждены гайдлайны, которые нормально подходят для CEO и букридеров;
* не используются или четко отбираются библиотеки и компоненты;
* все разработчики знают о том где какие заголовки используются и защищают код от внезапных поломок;
* дизайнеры тоже все это помнят;
* не надо вводить в проект новых людей или передавать проект на сторону;
* и нет риска глобального редизайна сотен страниц;

то можно оформлять теги заголовков. Допустим. Но плюсы-то в чем? Только в том чтобы уметь переназначать все стили заголовков в одном месте?

[](https://gist.github.com/murashki)

[**murashki**](https://gist.github.com/murashki)commented [on 10 Oct](https://gist.github.com/iAdramelk/d328b73c72cab92ef95f#gistcomment-1593032)

Но плюсы-то в чем? Только в том чтобы уметь переназначать все стили заголовков в одном месте?

Плюсы просто в том, что бы жить и заниматься любимым делом с умом )) Понимаете, вы не предлагаете по сути ничего кардинально нового. Вы предлагаете вариант, по сути дела ничем не отличающийся от того, что бы назначить стили дефолтным тегам. Просто теперь вам в довесок нужно будет постоянно добавлять класс к тегу. И все. Ваш метод все равно не даст возможности программистам делать их работу бездумно.

Обратите внимание что у вас в начале рассуждения было, и что в конце и скажите в чем отличие?

В начале была проблема:

Ты переназначил стили h1 и h2. Ну ой, теперь тебе надо протыкаться по 100+ типам страниц и убедиться что у тебя не сломалось ничего

Появилось решение:

Так что делать-то? Когда мы хотим поменять стили заголовка, мы создаем новый миксин, обходим вручную каждый блок где он используется и по очереди меняем и тестируем.

Чувствуете, что проблема не решена ни сколько? Вам так же нужно что-то обходить, что-то проверять и про что-то помнить и знать. Но еще у вас появилось куча ручной работы, куча классов. Осталась та же необходимость в проверке "все ли нормально". Единственное, что у вас получилось - это приобрести уверенность в том, что вы делаете все правильно. Вот и все.

Затрону тему все же про зеленую подложку и внезапно изменившийся дефолтный цвет заголовков. Смотрите, в бутстрапе есть дефолтный цвет текста. Есть также цветовые варианты для положек. Любой из этих вариантов очень гармонично смотрится с текстом в дефолтном цвете. Если вы изобретаете новую зеленую подложку для какого-то компанента, она должна быть добавлена в вашу цветовую палитру и должна сочетаться с текстом в дефолтном цвете как минимум. И уже далее, когда вы внезапно меняете цвет заголовков, ваш дизайнер должен убедиться в том, что новый дефолтный цвет по прежнему будет гармонично смотреться с любой подложкой, и если это не так, то нужно изменить варианты использующихся подложек. Вот решение. Не получится бездумно менять дефолтный цвет заголовков, но на это никто и не должен рассчитывать.

Вы же предлагаете мне отдуваться за действия бездумных программистов ))) Я так не играю.

[](https://gist.github.com/iAdramelk)

Owner

[**iAdramelk**](https://gist.github.com/iAdramelk)commented [on 10 Oct](https://gist.github.com/iAdramelk/d328b73c72cab92ef95f#gistcomment-1593191)

[@murashki](https://github.com/murashki) На самом деле нет. :) Я получил два больших плюса: 1) Стабильность работы сервиса – его теперь гораздо сложнее незаметно сломать. 2) Возможность тонкой настройки заголовков разных блоков под разные нужны и больший контроль над их изменениями.

И то, и то офигительные штуки. Добился я этого за счет договоренности о том что нет просто тегов или классов заголовком, а есть только классы заголовков отдельного блока, например:.review\_\_title. Я потерял в процессе возможность изменять все заголовки разом, но она мне не очень нужна на самом деле.

А в Амазон на Бутстрапе с маленьким набором стандартных элементов я не верю, извините уж.

[](https://gist.github.com/murashki)

[**murashki**](https://gist.github.com/murashki)commented [on 12 Oct](https://gist.github.com/iAdramelk/d328b73c72cab92ef95f#gistcomment-1593826)

Возможно ваш способ действительно хорош на больших проектах, но если это так, что уж наверняка для маленьких проектов он будет где-то избыточным. А раз так, то почему бы бутстрапу не быть?

[](https://gist.github.com/iAdramelk)

Owner

[**iAdramelk**](https://gist.github.com/iAdramelk)commented [on 12 Oct](https://gist.github.com/iAdramelk/d328b73c72cab92ef95f#gistcomment-1593863)

[@murashki](https://github.com/murashki) Бутстрапу конечно же быть. :) У него есть вполне понятная зона применения и назначение: CMS, CRM, прототипы, сайты опенсорсных проектов, которые собирают программисты из стандартных элементов, и т. д.

Главное не делать на нем сайтов которые вылазят за стандартный набор Бутстраповских элементов и быть готовым к тому, что когда (это важно, не если, а когда) они начнут за него выходить надо будет принять волевое решение – вместо того чтобы пытаться поддерживать Бутстрап, выкинуть его и переписать все по-человечески. А так да, до тех пор нормально.

[](https://gist.github.com/denji)

[**denji**](https://gist.github.com/denji)commented [on 17 Oct](https://gist.github.com/iAdramelk/d328b73c72cab92ef95f#gistcomment-1598585)

Google рекомендует БЭМ, селекторы по БЭМ имеют закономерное время отклика. Экономия потребления мобильных ресурсов. Особенно актуально для ожидаемого/среднего latency на выборку CSS-селекторов.

* [Google Developers – Ограничение объема и сложности вычисления стилей](https://developers.google.com/web/fundamentals/performance/rendering/reduce-the-scope-and-complexity-of-style-calculations#section-3)

В зависимости от используемого селектора и браузера на вычисление всего этого может уйти масса времени. Приблизительно **50% времени, которое тратится на вычисление стиля элемента, уходит на сопоставление селекторов**, а вторую половину времени занимает построение RenderStyle (представления стиля) на основе сопоставленных правил.

[](https://gist.github.com/levtolstoi)

[**levtolstoi**](https://gist.github.com/levtolstoi)commented [on 31 Oct](https://gist.github.com/iAdramelk/d328b73c72cab92ef95f#gistcomment-1609326)

Вообще не понимаю людей которые подключают бутстрап !целиком, зачем то потом пишут оверрайд(чего вообще не стоит делать), удивляются что у них что то сломалось, а потом начинают ругаться на бутстрап.