## Рекомендации html-academy:

* Мягкий валидатор: [**https://validator.w3.org/nu/**](https://validator.w3.org/nu/)
* [**http://instanceof.pro/code-guide/**](http://instanceof.pro/code-guide/) **- styleguide от bootstrap**
* Использовать методологию оформления: MOD (bootstrap), Google или БЭМ
* MOD: называть классы строго в соответствии с собственным стилем.

**top-nav\_\_nav-button--hidden**

* Использовать разбитие свойств в CSS по блокам: сначала позиционирования, потом размеры элемента, потом паддинг\маргин, границы, бекграунд, фонт
* Использовать псевдоэлементы **:before :after** для добавления элементов контента (ромбик)
* У всех изображений в теге img указывать размер
* Скрипты подключать в конце body
* Использовать минимально возможное количество html элементов
* Названия форм полей привязаны к своим полям с помощью label
* Не использовать вложенность селекторов больше 2
* **btn** - общий класс для стилизации кнопок
* **<button>** использовать только тогда, когда ссылка открывается на этой же странице, не ведет на другую
* **<a> -** кнопка купить товар, кнопка "в корзину", кнопка "вход"
* **<p>** и другие абзацы можно использовать тогда, когда текст имеет завершенную структуру. Например Контакты (телефон, адрес и т.д.) нельзя делить абзацами, можно <br> или <span>
* **Схлопывание элемента:** если у верхнего элемента margin-bottom: 20px, а у нижнего 40px, то margin между ними не складывается, а выбирается больший. Т.е. будет 40px
* **Выпадание внешних отступов:** если у родителя margin-top: 0, а у дочернего элемента margin-top: 40px, то отступ родителя будет тоже 40px. Что бы этого избежать, лучше использовать padding, либо задать границу родителю
* **clearfix или распорка:** поскольку сетка на float, к каждой колонке добавляем

.floating-block::after{

content: "";

display: table;

clear: both;

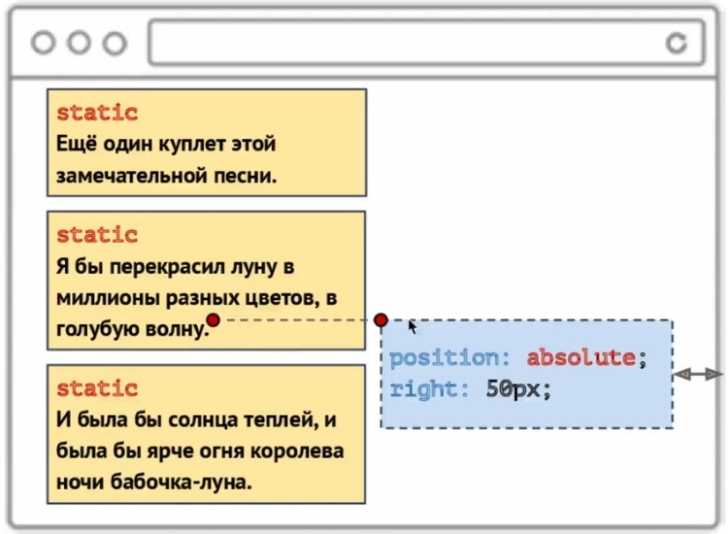
}

* **display: inline-block:** удобно делать мелкие элементы, например кнопку. Что бы не float-ить две кнопки рядом, можно им задать inline-block и задать padding. Борьба с отступами между инлайн-элементами: **htmlacademy.ru/blog/21**
* для выделения обьектов используют outline, background-color или box-shadow
* **inline-block:** если один блок в ряду **вылезает** за другой, скорее всего количество строк в блоках отличается, и выравниваются они по vertical-align: baseline.

**Исправить: задать vertical-align: top;**

* удобно делать сетку плавающих элементов, например как галерея на моем сайте - 3 ряда по 3 элемента. Выравнивание **vertical-align: center** позволит более-менее красиво их выровнять, даже если высота элементов отличается. Отличие от float: нет **зацепов за элементы**, которые отличаются по высоте
* **Пробелы между инлайн блоками** возникают, потому что каждый блок рассматривается как символ. Если верстать блоки без пробелов в html-коде (плохой код), пробелов не будет. Есть способы лучше:

- задать контейнеру **font-size: 0;** дочерним элементам (inline-block) - свой **font-size**

**position: absolute.** Если не указан явно top, абсолютный элемент привязывается левой верхней точкой к тому месту, откуда выпал. Работает стиль **top: auto** по умолчанию. Удобно использовать в разметке статей, что бы фотки располагались справа от абзаца. При смещении абзаца вниз-вверх, фото сдвигается автоматически. Так же, на эту точку действуют margin

* **мульти фон:** превое свойство выше второго, второе выше третьего и т.д.

background: url("../images/map\_pattern.png") 0 50px repeat-x, #dd3c26;

можно еще так:

background: #dd3c26 url("../images/map\_pattern.png") 0 50px repeat-x;

второй вариант без запятой

* **Высота строки: font 14px/24px.** Из высоты строки вычитается фон, оставшееся место делится пополам. т.е. тут отступы сверху и снизу будут по 5px. Как паддинг и строка внутри
* **button и input - текстовые кнопки (inline).** Именно потому у button текст всегда по центру если у кнопки есть ширина, задавая padding, она не увеличится. Т.е. кнопка как box-sizing: padding-box
* **Для картинок задавать фон, похожий по цвету.** Так же задавать похожий по цвету фон для всех фонов - хедеры, футеры и т.д. Для img задавать размер, как страховка от того, что юзер вставит большую картинку
* **Сделать треугольник на CSS:**

.test{

width: 0;

C:\Users\Alexandr\Downloads\triangle.JPG height: 0;

border: 20px solid orangered;

border-left-color: green;

}

* **Закон близости контента**: к заголовку относится та часть контента, которая к нему ближе. Потому для заголовков обычно задают margin-top в два раза больше margin-bottom. Для обычных элементов задают только margin-bottom
* **margin не выпадает из float-контейнера**. Это значит , что маргин нужно задавать самой обертке, а не его дочерним элементам
* **<label> НЕ вкладывать в <input>, а писать после него**: нужно для того, что бы работал прием стилизации checkbox:

.login-form input[type="checkbox"] { display: none}

.login-form input[type="checkbox"] + label::before{

content: ""; background: url("icon")

}

* **Тестирование:** верстка проходит тест на переполнение контентом:
* не ломается при добавлении в элементы большего количества текста;
* Не ломается при использовании картинок с неподходящими размерами
* Не ломается при изменении количества потоковых блоков
* **Прогрессивное улучшение - отказоустойчивость:**
* От простого к сложному
* На любом этапе должен получится законченный веб-интерфейс
* Улучшения добавляются слоями
* Слои должны быть цельными или "атомарными"
* Новые слои не ломают предыдущие

## Рекомендации чувака с хабра

## С чего начать: Выучить HTML\CSS Зарегистрироваться на odesk.com там очень хорошие тесты. По ним можно проверять знания в CSS. Вот правила для моих верстальщиков и всех кто работает со мной на проектах, можно многое подчеркнуть для начала. [https://docs.google.com/document/d/1kehaJKKRo7zxYp...](https://docs.google.com/document/d/1kehaJKKRo7zxYpwjPz8-iQD0jVAYUHP-G96yrl04h5Q/edit?usp=sharing) список блогов для чтения [https://www.dropbox.com/s/chwesp5gh5rths2/feedly.o...](https://www.dropbox.com/s/chwesp5gh5rths2/feedly.opml?dl=0) НЕ НАЧИНАЙТЕ ИЗУЧЕНИЕ С БУТСТРАПОВ И ТД, Что первое учить: 1. Блочная модель 2."Поток" - позиционирование элементов (абсолюты, релятивы...) 3. Плавающие элементы (float) 4. Верстка форм (как со стороны html\css) 5. Понять методологию BEM и научиться использовать принципы. 6. Изучить Jade/SASS(можно LESS или Stylus, но опыт показывает что SASS круче) 7. HTML5 - (сначала блоки верстать используя только div, как толкьо появиться "чувство" структуры информации можно лезть в дебри использования HTML5 тегов) 8. CSS3 - трансформации, анимация, 3d 9. Flex модель (учить только опосле полного понимания плавающих элементов и их вариаций) ... P.S. Книги читать не советую, лучше прочитать весь хаб по верстке на хабре, будет больше пользы. Знания и умение правильно верстать приходят только с практикой так что стандартный путь по которому прошел я сам это: ищем макет (биржы фриланса, бесплатные шаблоны...) верстаем его loop... если верстать каждый день и читать не менее 50 страниц тематической литературы в день, через 6 месяцев вы сможете верстать лучше чем 50% на рынке...)

## <https://toster.ru/q/89477>

## ПРАВИЛА ВЕРСТАЛЬЩИКОВ

## Основные

1. Соответствие макету
2. Кроссбраузерность (ie9+, последние версии Chrome, Mozilla, Opera)
3. Валидность:
4. Стили сайта соответствуют стандарту CSS3, за исключением хаков и вендорных свойств.
5. Верстка сайта соответствует стандарту HTML5 за исключением уникальных свойств браузеров и результатов выдачи WYSIWYG-редакторов.
6. Не должно быть JavaScript-ошибок.
7. Кодировка: UTF-8.
8. DOCTYPE: HTML5
9. CSS и JavaScript в отдельных файлах.
10. Не должно быть js-ошибок!
11. CSS3 border-radius, gradient, box-shadow, text-shadow вместо использования графики.
12. Обязательно использование препроцессора sass в форме scss.
13. Все поля, содержимое которых приходит из CMS проверены на работу с разным вариантом контента: короткий, длинный, длинный без переноса.
14. Обязательное наличие label для input. Поля label и input/select в форме слинкованы.
15. Независимость блоков в CSS: минимизация каскада, использование техник БЭМ
16. Сайт должен нормально смотреться во всех стандартных разрешениях от 1024 и выше, не иметь горизонтального скролла и вписываться в экран мобильных устройств
17. Корректная работа при вбивании реального текста, надёжность вёрстки
18. Проверка и оптимизация скорости загрузки (оптимизация изображений; html, css-кода; количества запросов к серверу)
19. Правильная структура заголовков (H1,H2,… и т.д. и TITLE)

Доступность

1. выделяется ли текст в текстовых блоках; кликаются ли кликабельные элементы (ссылки/кнопки);
2. устанавливается ли фокус в поля форм;
3. кликабельные элементы должны иметь указатель "курсор", перетаскивающиеся - "лапка" или "ресайз", активные/недоступные - курсор default;
4. в идеале все активные элементы должны реагировать на наведение, не доступные/неактивные - не должны;
5. клик по label должен переводить фокус в связанное поле;
6. кликабельные элементы, назначение которых не очевидно должны быть снабжены подсказками (title);

Расширяемость

1. увеличиваем/уменьшаем к-во контента для блоков где это может произойти (текст2.овые блоки, меню, списки чего-либо и т.п.);
2. вбиваем реальный контент с тегами в блоки, где такое может быть (проверка контента по шаблону);

Соответствие стандартам.

1. Для проверки надежней использовать валидаторы на W3C (HTML и CSS).
2. HTML не должен содержать ошибок/предупреждений. В качестве исключений можно оставить обоснованные ошибки (например, target=\_blank);
3. в CSS обращаем внимание только на синтаксические ошибки;
4. в CSS не должны присутствовать правила и хаки для IE (такие должны быть вынесены в отдельные стилевые файлы);

Дополнительные

Файловая структура

project\_name/

i/

js/

fonts/

styles/

sass/

index.html

Разное

* footer всегда прижат к низу экрана
* Для свойств margin и padding всегда использовать короткую запись.
* Обозначение цвета либо в hex (#000), либо в rgba если есть прозрачность, никаких ключевых слов.
* Всегда задавать значения для позиционирования бэкграунда(background-position), и задавать их без ключевых слов.
* при верстке не использовать id, только классы
* все оформительские изображения делаем через background
* использовать сброс стилей по Э. Мейеру (reset)
* разумная минимизация html-кода, вместо элементов для стилизации - использовать :before и :after.
* для слайдеров использовать bxSlider, для стилизации элементов форм - chosen.js
* изображение(img) всегда должно иметь элемент обертку, с указанием для этой обертки размеров и overflow:hidden;

Именование классов:

* только маленькие буквы
* если класс из нескольких слов – между словами подчеркивание: slider\_block, bread\_crumbs и т.п.,
* если нужно добавить обработчик js к какому то элементу ему даеться дополнительный класс в "кэмел стайл" <http://ru.wikipedia.org/wiki/CamelCase> .
* без дефисов, без camelCase.
* все теги должны иметь класс. Исключение: элементы которые содержатся в блоке, содержимое которого наполняется контент-менеджером. Как правило, такой блок должен иметь класс "static"
* ознакомиться с технологией БЭМ от Яндекса, для понимания принципов именования классов.

Особенности блочности и модульности элементов

* Модификации и любые классы состояний для блочных элементов и их составляющих называть с именованием класса "соответствия"

***пример:***

*родительский класс "close\_butt"*

*иконки "close\_b\_ico\_1"*

*размера "close\_b\_mod\_v1"*

*цветовой схемы "close\_b\_grey\_blue\_mod"*