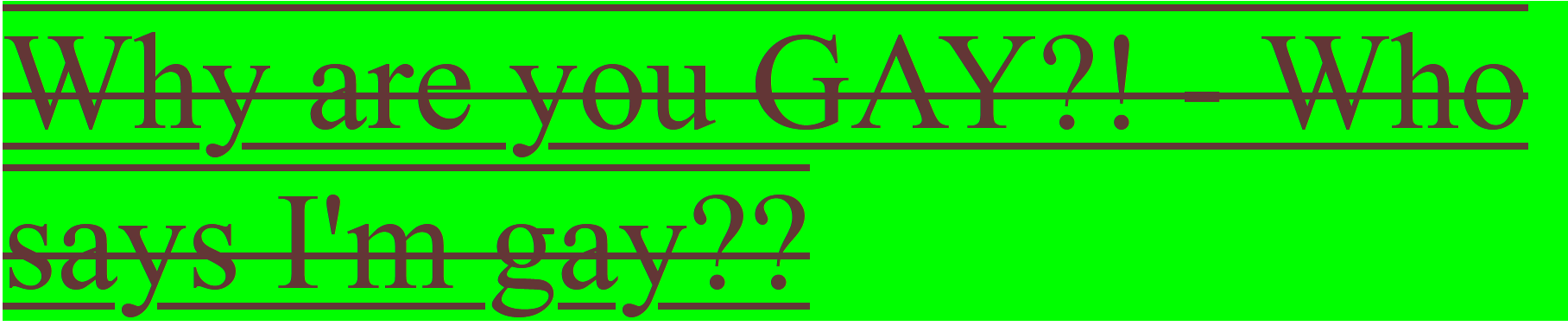
Несколько значений свойства указываются через пробел

text-decoration: overline underline line-through;



**Как задавать цвет**

**HEX -** #FFFFFF (RRGGBB) – 3 шеснадцатиричных числа, которые задают уровень красного, зеленого и синего

Возможно сокращение до 3х символов, если они повторяются парами #FFF

**RGB**

rgb(255, 0, 0)

rgb(100%, 50%, 50%).

rgba(255, 255, 255, 0.5) – с альфа каналом (прозрачность) 0-полностью прозрачен, 1 непрозрачен

**HSL (hue - оттенок, saturate - насыщенность, lightness - светлость)**

H(0-359) – определяется на цветовом круге, задается в градусах. 0° соответствует красному цвету, 120° — зелёному, а 240° — синему

S(0-100%) - Значение 0% обозначает отсутствие цвета и оттенок серого, 100% максимальное значение насыщенности.

L(0-100%) - Малые значения делают цвет темнее, а высокие светлее, крайние значения 0% и 100% соответствуют чёрному и белому цвету.

Hsl(240, 50, 50)

Hsla(240, 50, 50, 0.5) – с альфа-каналом (прозрачность)

**Через названия** (Red, blue, green)

**РАЗМЕРЫ**

**Абсолютные** –

* px (пиксели)
* pk
* cm (сантиметр) = 38px – раньше использовали для версий печати.
* mm (миллиметр) = 3.8px
* dm

**Относительные** –

* **%** - определение размеров относительно размеров родительского блока, либо величин заданных браузером.
* **Rem** – root element (вычисляется от размера шрифта корневого элемента html, если не указано – относительно браузера. Для использования rem можно установить базовый размер шрифта, а затем применять rem для определения цветов относительно базового размера.),
* **em** – относительно шрифта родителя. 1em – это размер шрифта родительского блока.
* **Vh** – высота в процентах относительно viewport (размеров области просмотра), 1vh = 1% от высоты окна.
* **Vw** – ширина в процентах от экрана, 1 vw = 1% от высоты окна.
* **vmin** – использует размер меньшей стороны, то есть если высота окна браузера меньше ширины, 1 vmin=1vh. Если ширина меньше высоты, 1vmin=1vw.
* **vmax** – используют большую сторону.

Исключения!

*При установке свойства margin и padding в % процент берется от ширины родительского блока, даже если задает вертикальные отступы.*

*При установке свойства line-height в % процент берется от текущего размера шрифта, а вовсе не от line-height родителя.*

*Для width/height процент берется обычно от ширины/высоты родителя, но при position: fixed процент задается от ширины/высоты окна (а не родителя или документа).*

Базовым размером шрифта, который по умолчанию применяется большинством браузеров является 16px.

Для определения **размера шрифта** css также предлагает 7 ключевых слов, которые задают размер относительно базового:

* xx-small
* x-small
* small
* medium (базовый 16px)
* large
* x-large
* xx-large

**СВОЙСТВА ДЛЯ ТЕКСТА**

* **font-family –** шрифт:Times New Roman, Georgia, serif, sans-serif
* **font-size –** размер шрифта, **inherit –** наследование от родителя, **initial** – по умолчанию.
* **font-weight** – жирность: 100-200-900, normal – 400, bold – 700, initial, inherit.
* **font**-**style –** стиль шрифта. normal, italic(курсив)
* **letter**-**spacing –** расстояние между буквами: normal, 10px, можно отрицательные -10px, em.
* **word-break** - переносы: normal(стандартный), break-all(при превышении пойдет перевод, между символами) break-word(словами)

**text-decoration (доп оформление):**

* **underline** –подчеркнутый текст внизу(пример)
* **overline** – подчеркнутый текст сверху (пример)
* **line-through** – перечеркнутый текст (~~пример~~)
* **none** – убирает подчёркивание (у ссылок)

**Text-transform(работа с регистрами):**

* **lowercase –** символы текста строчные (нижний регистр).
* **uppercase** – символы текста прописные (верхний регистр).
* **white-space(**обработка пробелов): normal(несколько в один), no-wrap(без переноса)
* **none** – не меняет регистр символов.

**Text-align(выравнивание)**

• left/center/right/justify

**line-height - межстрочное расстояние.**

**Text-indent(ОТСТУП),** ПО ДЕФОЛТУ ТОЛЬКО ПЕРВАЯ СТРОЧКА,

* text-indent: 5em each-line;(ВСЕ СТРОКИ ПОСЛЕ ENTER И BR)
* text-indent: 5em hanging; (ВАЩЕ ВСЕ СТРОКИ КРОМЕ ПЕРВОЙ)

**CЕЛЕКТОРЫ**

Селекторы могут **группироваться** в одной строке через запятую. Тогда стили применяются к каждому из них.

**Виды селекторов:**

* *\* {}* универсальный селектор. Правило применяется к любому элементу. Может использоваться в комбинации с псевдоклассами и псевдоэлементами.
* *p {}* селектор элемента. Применяется ко всем элементам данного типа.
* *.class {}* селектор класса. Применяются к элементам с совпадающим атрибутом class. Если элемент имеет несколько классов, то они разделяются пробелом.
* *#id {}* селектор идентификатора. Применяется к элементу с указанным идентификатором.
* *ul li {}* селектор потомков. Применяется к элементам, расположенным внутри элемента контейнера.
* *p > b* дочерний селектор. Применяется ко всем элементам, являющимися прямыми потомками контейнера.
* *h1 + p* сестринский селектор. Сестринские элементы – те которые имеют общего родителя. Выбирается при этом только первый элемент, следующий после h1.

*h1 ~ p* выберет все сестринские элементы, следующие за h1.

* *a:active* селектор псевдокласса. позволяют применить CSS-правила к элементам при совершении события или подчиняющимся определенному правилу.
* *p::first-letter* селектор псевдоэлемента.

**Селекторы атрибутов:**

* *p[attribute=”value”]* селектор атрибута. Выбирает элементы на основе имени атрибута или его значения
* *p[attr] –* элемент с указанным атрибутом
* *p[attr~=value] –* значением аттрибута является набор слов разделенных пробелами, одно из которых в точности равно value
* *[attr|=value]* - значение атрибута может быть или в точности равно "value" или может начинаться с "value" со сразу же следующим "-"
* *[attr^=value]* - значение атрибута начинается с "value"
* *[attr$=value]* – значение атрибута заканчивается на "value"
* *[attr\*=value]* – значение атрибута содержит по крайней мере одно вхождение строки "value" как подстроки.
* *[attr1=value1][attr2^=value2]* – селектор по нескольким атрибутам.

**Псевдоклассы** определяют динамическое состояние элементов, которое изменяется с помощью действий пользователя, а также положение в дереве документа.

Условно все псевдоклассы делятся на три группы:

* **определяющие состояние элементов;**

1. **active** – активация элемента (щелчок мыши);
2. **link** – непосещенная ссылка;
3. **focus** – элемент получивший фокус (например курсор в поле для ввода);
4. **hover** – при наведении мышки на элемент;
5. **visited** – посещенная ссылка;

* имеющие отношение к дереву элементов;

1. **first-child** – первый дочерний элемент селектора.

* указывающие язык текста.
  1. **lang()** – определяет язык, который используется в документе или его фрагменте.

**Псевдоэлемент** в CSS — это ключевое слово, добавляемое к селектору, которое позволяет стилизовать определённую часть выбранного элемента.

* 1. **after** – применяется для вставки контента после содержимого элемента. Вставляемый контент определяется свойством content в css правиле.
  2. **before** – для вставки контента до содержимого элемента.
  3. **first-letter** – определение стиля первого символа в тексте.
  4. **first-line** – стиль для первой строки блочного текста.

**Наследование –** механизм, с помощью которого значения свойств элемента-родителя передаются его элементам-потомкам. Потомки могут переопределять свой стиль.

Не все свойства наследуются. Чтобы унаследовать ненаследуемые свойства используется слово inherit. Чтобы запретить наследование используется *initial*. В этом случае элемент применит свойства браузера.

*Наследуемые свойства:* отображающее параметры текста (fon-size, font-family, color, text-align, direction), cursor, visibility, list-style

**Каскадирование стилей –** заключается в том, что если к элементу применяется несколько css-правил, то результирующий стиль будет сформирован из всех присутствующих в этих правилах свойств.

В случае конфликтов, будет применено свойство с наибольшим приоритетом.

* + 1. Стиль браузера;
    2. Стиль пользователя (устанавливается в настройках браузера).
    3. Стиль автора страницы

Стили унаследованные от родителей.

Стили заданные во внешних таблицах.

Стили внутренних таблиц в атрибуте <style>.

Инлайн-стили.

Стили помеченные !important.

В случае конфликта на одном уровне, конфликт решается с помощью сравнения специфичности селекторов. Приоритетнее те, что имеют больше

1. идентификаторов;
2. затем классов, атрибутов и псевдоклассов;
3. элементов и псевдоэлементов.

В случае одинаковой специфичности, применяется стиль, расположенный ниже в таблице.

**Методология –** совокупность принципов, способов и средств применяющихся для написания кода, и имеющих общий философский подход.

**БЭМ**

Методология БЭМ состоит в организации css-кода в повторно используемые модули с помощью умной системы именования.

Используются для:

* Устранения дублирования кода, любой блок может быть повторно использован.
* Устранение взаимного влияния компонентов друг на друга. Грубо говоря изменяя что-то в одной части, можешь быть уверен, что это не сломает что-то еще.
* Самодокументирующийся код. Помогает при переходе на новый проект.

**БЛОК (что это?) -** Логически и функционально независимый компонент страницы. Блок полностью самодостаточен.

Независимость блоков обеспечивает возможность их повторного использования.

Блоки можно вкладывать в любые другие блоки.

Блоки можно перемещать в пределах одной страницы и разных проектов. Независимость блока позволяет изменять его положение на странице и обеспечивает корректную работу и внешний вид.

Блоку не рекомендуется устанавливать в стилях внешнее позиционирование. Чтобы мы могил повторно его использовать.

**Элемент (что это?) –** составная часть блока, которая не используется вне самого блока. Элемент имеет смысл только в рамках своего родителя. Например, пункт меню вне блока меню не используется, значит является его элементом.

Работая с элементами, важно помнить правило: не рекомендуется создавать элементы элементов. Если вложить один элемент в другой, будет невозможно изменить внутреннюю структуру блока: элементы нельзя будет поменять местами, удалить или добавить без корректировки существующего кода.

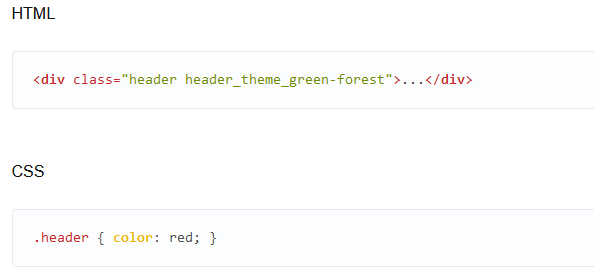
**Модификатор (какой, как выглядит, как себя ведет, какое состояние?) -** БЭМ-сущность, определяющая внешний вид, состояние и поведение блока или элемента. Использование модификаторов опционально, количество — неограниченно. Блоку или элементу нельзя одновременно присвоить разные значения модификатора. Модификатор имеет имя и может иметь значение.

**Правила формирования имени БЭМ-сущности**

* Каждая БЭМ-сущность должна иметь свой класс.
* CSS-свойства для блоков, элементов и модификаторов описываются только через классы.
* Для разделения слов в именах используется дефис (-).
* Элемент отделяется от блока двумя подчеркиваниями (\_\_). Модификатор — одним (\_).
* Имена БЭМ-сущностей записываются с помощью цифр и латинских букв в нижнем регистре.

**Пример**

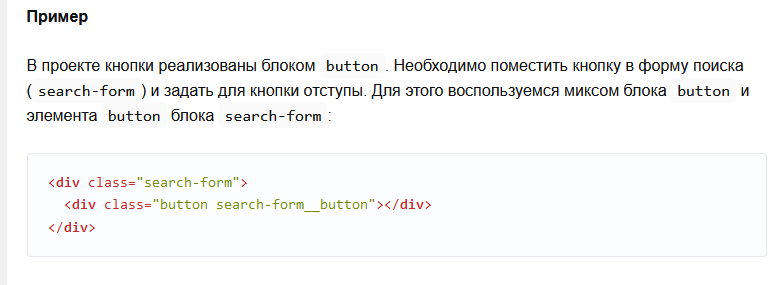
* **Имя блока** — header.
* **Имя элемента блока** — header\_\_search-form — элемент search-form блока header.
* **Имя модификатора блока** — header\_theme\_green-forest — **модификатор** theme в значении green-forest блока header.
* **Имя модификатора элемента** — header\_\_search-form\_disabled — булев модификатор disabled элемента search-form блока header.



**Микс -** способ использования разных БЭМ-сущностей на одном DOM-узле.

Миксы позволяют:

* объединять поведение и стили нескольких БЭМ-сущностей без дублирования кода;
* создавать семантически новые компоненты интерфейса на основе имеющихся блоков, элементов и модификаторов;
* задавать позицию вложенного блока в родительском, не создавая дополнительных модификаторов (То есть сам вид компонента описан как блок, миксуем его с элементом. В элементе уже указываем его позиционирование).



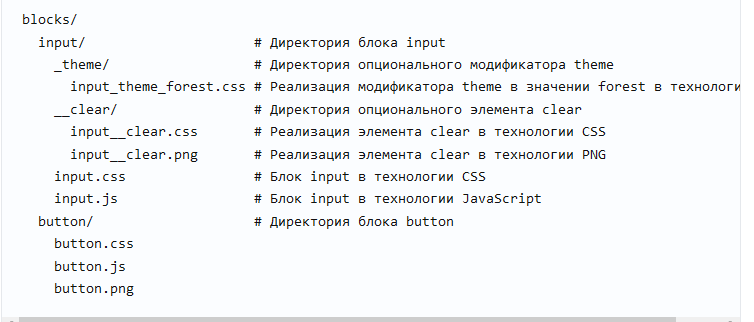
**Структура папок**

**Блоки первичны, технологии вторичны.**

* Блок — отдельная директория в файловой системе. Имя блока и его директории совпадают.
* Реализация блока разделяется на отдельные файлы.
* Файлы, относящиеся к блоку, всегда находятся в его директории.
* Опциональные элементы и модификаторы выносятся в отдельные файлы.
* Проект разделяется на [уровни переопределения](https://ru.bem.info/method/key-concepts/#Уровень-переопределения).

Блоки могут быть реализованы в одной или нескольких технологиях, например:

* поведение — JavaScript, CoffeeScript;
* внешний вид — CSS, Stylus, Sass;
* шаблоны — Pug, Handlebars, XSL, BEMHTML, BH;
* документация — Markdown, Wiki, XML.



**ATOMIC**

Модель построения CSS-кода, при которой почти каждому CSS свойству соответствует свой класс.

1. Универсальность имен – имя не несет никакого семантического значения, а лишь css значение и свойство.
2. Читабельность классов – если человек видит класс с названием width100px, он сразу поймет что он задает, не смотря в код.
3. Универсальность классов – один и тот же класс может быть использован для различных html элементов, при этом их можно комбинировать различным образом.

**Enduring CSS** (выносливый CSS) или **eCSS –** центральной концепцией является изоляция. Все компоненты представляют собой изолированную единицу кода, без зависимостей. Эту единицу можно использовать повторно и удалять без риска утечки стилей.

Достигается за счет:

* инкапсуляции всего кода, не только CSS, а всех технологий, необходимых для постройки всех компонентов в своих общих папках;
* создания совершенно нового компонента каждый раз, когда нужен компонент, похожий на уже существующий, но в котором будут некоторые изменения, даже если изменения незначительны;
* использования строгой системы именования.

Что вы получите, если примете методологию eCSS и закроете глаза на повторения:

* изоляция всех визуальных шаблонов облегчает обслуживание CSS-кода;
* несмотря на повторяющиеся свойства и значения, размер файла в долгосрочной перспективе почти не увеличится. Это происходит потому, что модули являются автономными, изолированными единицами, которые можно быстро удалить без страха поломать дизайн, если они больше не нужны;
* все языковые/технические файлы, необходимые для создания модуля, расположены в одной папке, что сильно упрощает редактирование и физическое удаление.

**ДРУГОЕ**

**position –** css свойство, которое устанавливает способ позиционирования элемента относительно окна браузера или других объектов на веб-странице.

Значения:

**absolute**

Элемент исчезает с того места, где он должен быть и позиционируется заново. Остальные элементы, располагаются так, как будто этого элемента никогда не было. Координаты top/bottom/left/right для нового местоположения отсчитываются от ближайшего позиционированного родителя, т.е. родителя с позиционированием, отличным от static. Если такого родителя нет – то относительно документа.

**fixed**

По своему действию это значение близко к absolute, но в отличие от него привязывается к указанной свойствами left, top, right и bottom точке на экране и не меняет своего положения при прокрутке веб-страницы. Браузер Firefox вообще не отображает полосы прокрутки, если положение элемента задано фиксированным, и оно не помещается целиком в окно браузера. В браузере Opera хотя и показываются полосы прокрутки, но они никак не влияют на позицию элемента.

**relative**

Положение элемента устанавливается относительно его исходного места. Добавление свойств left, top, right и bottom изменяет позицию элемента и сдвигает его в ту или иную сторону от первоначального расположения.

**static (по умолчанию)**

Элементы отображаются как обычно. Использование свойств left, top, right и bottom не приводит к каким-либо результатам.

**inherit**

Наследует значение родителя.