**CSS и переменные**

**CSS** ([/siːɛsɛs/](https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D0%B6%D0%B4%D1%83%D0%BD%D0%B0%D1%80%D0%BE%D0%B4%D0%BD%D1%8B%D0%B9_%D1%84%D0%BE%D0%BD%D0%B5%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%B0%D0%BB%D1%84%D0%B0%D0%B2%D0%B8%D1%82) [англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *Cascading Style Sheets* — *каскадные таблицы стилей*) — [формальный язык](https://ru.wikipedia.org/wiki/%D0%A4%D0%BE%D1%80%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) описания внешнего вида документа, написанного с использованием [языка разметки](https://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D1%80%D0%B0%D0%B7%D0%BC%D0%B5%D1%82%D0%BA%D0%B8).

Преимущественно используется как средство описания, оформления внешнего вида [веб-страниц](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D1%82%D1%80%D0%B0%D0%BD%D0%B8%D1%86%D0%B0), написанных с помощью [языков разметки](https://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D1%80%D0%B0%D0%B7%D0%BC%D0%B5%D1%82%D0%BA%D0%B8) [HTML](https://ru.wikipedia.org/wiki/HTML) и [XHTML](https://ru.wikipedia.org/wiki/XHTML), но может также применяться к любым [XML-документам](https://ru.wikipedia.org/wiki/XML), например, к [SVG](https://ru.wikipedia.org/wiki/SVG) или [XUL](https://ru.wikipedia.org/wiki/XUL).

**Цель создания CSS**

CSS используется создателями [веб-страниц](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D1%82%D1%80%D0%B0%D0%BD%D0%B8%D1%86%D0%B0) для задания [цветов](https://ru.wikipedia.org/wiki/%D0%A6%D0%B2%D0%B5%D1%82), [шрифтов](https://ru.wikipedia.org/wiki/%D0%A8%D1%80%D0%B8%D1%84%D1%82), расположения отдельных блоков и других аспектов представления внешнего вида этих веб-страниц. Основной целью разработки CSS являлось разделение описания логической структуры веб-страницы (которое производится с помощью [HTML](https://ru.wikipedia.org/wiki/HTML) или других [языков разметки](https://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D1%80%D0%B0%D0%B7%D0%BC%D0%B5%D1%82%D0%BA%D0%B8)) от описания внешнего вида этой веб-страницы (которое теперь производится с помощью [формального языка](https://ru.wikipedia.org/wiki/%D0%A4%D0%BE%D1%80%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) CSS). Такое разделение может увеличить доступность документа, предоставить большую гибкость и возможность управления его представлением, а также уменьшить сложность и повторяемость в структурном содержимом. Кроме того, CSS позволяет представлять один и тот же документ в различных стилях или [методах](https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4) вывода, таких как экранное представление, печатное представление, чтение голосом (специальным голосовым браузером или программой чтения с экрана), или при выводе устройствами, использующими [шрифт Брайля](https://ru.wikipedia.org/wiki/%D0%A8%D1%80%D0%B8%D1%84%D1%82_%D0%91%D1%80%D0%B0%D0%B9%D0%BB%D1%8F).

**Способы подключения CSS к документу**

Правила CSS пишутся на [формальном языке](https://ru.wikipedia.org/wiki/%D0%A4%D0%BE%D1%80%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) CSS и располагаются в таблицах стилей, то есть таблицы стилей содержат в себе правила CSS. Эти таблицы стилей могут располагаться как в самом [веб-документе](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D0%B4%D0%BE%D0%BA%D1%83%D0%BC%D0%B5%D0%BD%D1%82), внешний вид которого они описывают, так и в отдельных [файлах](https://ru.wikipedia.org/wiki/%D0%A4%D0%B0%D0%B9%D0%BB), имеющих [формат](https://ru.wikipedia.org/wiki/%D0%A4%D0%BE%D1%80%D0%BC%D0%B0%D1%82_%D1%84%D0%B0%D0%B9%D0%BB%D0%B0) CSS. (По сути, формат CSS — это обычный [текстовый файл](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BA%D1%81%D1%82%D0%BE%D0%B2%D1%8B%D0%B9_%D1%84%D0%B0%D0%B9%D0%BB). В файле .css не содержится ничего, кроме перечня правил CSS и комментариев к ним.)  
То есть, эти таблицы стилей могут быть подключены, внедрены в описываемый ими [веб-документ](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D0%B4%D0%BE%D0%BA%D1%83%D0%BC%D0%B5%D0%BD%D1%82) четырьмя различными способами:

* когда таблица стилей находится в отдельном файле, она может быть подключена к веб-документу посредством [тега](https://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%B5%D0%BC%D0%B5%D0%BD%D1%82%D1%8B_HTML) <link>, располагающегося в этом документе между тегами<head> и </head>. (Тег <link> будет иметь атрибут href, имеющий значением адрес этой таблицы стилей). Все правила этой таблицы действуют на протяжении всего документа;

<!DOCTYPE html>

<html>

<head>

.....

<link rel="stylesheet" href="style.css">

</head>

<body>

.....

</body>

</html>

* когда таблица стилей находится в отдельном файле, она может быть подключена к веб-документу посредством директивы @import, располагающейся в этом документе между тегами <style></style> (которые, в свою очередь, располагаются в этом документе между тегами <head></head>) сразу после тега <style>, которая также указывает (в своих скобках, после слова url) на адрес этой таблицы стилей. Все правила этой таблицы действуют на протяжении всего документа;

<!DOCTYPE html>

<html>

<head>

.....

<style media="all">

@import url(style.css);

</style>

</head>

</html>

* когда таблица стилей описана в самом документе, она может располагаться в нём между тегами <style> и </style> (которые, в свою очередь, располагаются в этом документе между тегами <head> и </head>). Все правила этой таблицы действуют на протяжении всего документа;

<!DOCTYPE html>

<html>

<head>

.....

<style>

body {

color: red;

}

</style>

</head>

<body>

.....

</body>

</html>

* когда таблица стилей описана в самом документе, она может располагаться в нём в теле какого-то отдельного тега (посредством его атрибута style) этого документа. Все правила этой таблицы действуют только на содержимое этого тега.

<!DOCTYPE>

<html>

<head>

.....

</head>

<body>

<p style="font-size: 20px; color: green; font-family: arial, helvetica, sans-serif">

.....

</p>

</body>

</html>

В первых двух случаях говорят, что к документу применены *внешние таблицы стилей*, а во вторых двух случаях — *внутренние таблицы стилей*.

Для добавления CSS к [XML-документу](https://ru.wikipedia.org/wiki/XML), последний должен содержать специальную ссылку на таблицу стилей. Например:

<?xml-stylesheet type="text/css" href="style.css"?>

### Иерархия элементов внутри документа

Как известно, [HTML-документы](https://ru.wikipedia.org/wiki/HTML) строятся на основании иерархии элементов, которая может быть наглядно представлена в древовидной форме. Элементы HTML друг для друга могут быть *родительскими, дочерними, элементами-предками, элементами-потомками, сестринскими*.

Элемент является *родителем* другого элемента, если в иерархической структуре документа он находится сразу, непосредственно над этим элементом. Элемент является *предком* другого элемента, если в иерархической структуре документа он находится где-то выше этого элемента.   
Пускай, например, в документе присутствуют два абзаца **p**, включающие в себя шрифт с полужирным начертанием **b**. Тогда элементы **b** будут *дочерними элементами* своих *родительских элементов* **p** и *потомками* своего *предка* **body**. В свою очередь, для элементов **p** элемент **body** будет являться только *родителем*. И кроме того, эти два элемента **p** будут являться *сестринскими элементами*, как имеющие одного *родителя* — **body**.

В CSS могут задаваться при помощи [селекторов](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D0%BB%D0%B5%D0%BA%D1%82%D0%BE%D1%80) не только одиночные элементы, но и элементы, являющиеся потомками, дочерними или сестринскими элементами других элементов (см. подраздел «виды селекторов»).

### Правила построения CSS

В первых трёх случаях подключения таблицы CSS к документу (см. выше) каждое правило CSS из таблицы стилей имеет две основные части — *селектор* и *блок объявлений*. *Селектор*, расположенный в левой части правила, определяет, на какие части документа распространяется правило. *Блок объявлений* располагается в правой части правила. Он помещается в фигурные скобки, и, в свою очередь, состоит из одного или более *объявлений*, разделённых знаком «;».

Каждое *объявление* представляет собой сочетание *свойства CSS* и *значения*, разделённых знаком ": ". Селекторы могут группироваться в одной строке через запятую. В таком случае свойство применяется к каждому из них.

селектор, селектор {

свойство: значение;

свойство: значение;

свойство: значение;

}

В четвёртом случае подключения таблицы CSS к документу (см. список) правило CSS (являющееся значением атрибута style тега, на который оно действует) представляет собой перечень *объявлений* («*свойство CSS* : *значение*»), разделённых знаком «;».

#### Виды селекторов

##### Универсальный селектор

\* {

margin: 0;

padding: 0;

}

##### Селектор тегов

p {

font-family: arial, helvetica, sans-serif;

}

##### Селектор классов

.note {

color: red;

background-color: yellow;

font-weight: bold;

}

##### Селектор идентификаторов

#paragraph1 {

margin: 0;

}

##### Селектор атрибутов

a[href="http://www.somesite.com"] {

font-weight: bold;

}

##### Селектор потомков (контекстный селектор)

div#paragraph1 p.note {

color: red;

}

##### Селектор дочерних элементов

p.note > b {

color: red;

}

##### Селектор сестринских элементов

h1 + p {

font-size: 24px;

}

##### Селектор псевдоклассов

a:active {

color: blue;

}

##### Селектор псевдоэлементов

p::first-letter {

font-size: 32px;

}

### Классы и идентификаторы элементов

Класс или идентификатор может быть присвоен какому-нибудь элементу (тегу) HTML посредством атрибутов class или id этого элемента (тега):

<!DOCTYPE html>

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8">

<title>

Селекторы классов и идентификаторов

</title>

<style>

p.Big {

font-family: arial, helvetica, sans-serif;

color: maroon;

}

div#First {

background-color: silver;

}

</style>

</head>

<body>

.....

<div id="First">

.....

</div>

<p class="Big">

.....

</p>

</body>

</html>

Основное отличие между классами элементов и идентификаторами элементов в том, что идентификатор предназначен для одного элемента, тогда как класс обычно присваивают сразу нескольким. Тем не менее, современные браузеры, как правило, корректно отображают множественные элементы с одинаковым идентификатором. Также отличие в том, что могут существовать множественные классы (когда класс элемента состоит из нескольких слов, разделённых пробелами). Для идентификаторов такое невозможно.

Важно отметить следующее отличие идентификатора от класса: идентификаторы широко используются в [JavaScript](https://ru.wikipedia.org/wiki/JavaScript) для нахождения уникального элемента в документе.

Имена классов и идентификаторов, в отличие от названий тегов и их атрибутов, чувствительны к регистру ввода букв.

Свойства классов и идентификаторов задаются с помощью соответствующих селекторов. Причём может быть задано как свойство класса в целом (в таком случае селектор начинается с «.») или свойство идентификатора самого по себе (в таком случае селектор начинается с «#»), так и свойство какого-нибудь элемента этого класса или с этим идентификатором.

В CSS помимо классов, задаваемых автором страницы, существует также ограниченный набор так называемых *псевдоклассов*, описывающих вид гиперссылок с определённым состоянием в документе, вид элемента, на котором находится фокус ввода, а также вид элементов, являющихся первыми дочерними элементами других элементов. Также в CSS существует четыре так называемых *псевдоэлемента*: первая буква, первая строка, применение специальных стилей до и после элемента.

### Наследование. Каскадирование. Приоритеты стилей CSS.

Применение CSS к документам HTML основано на принципах *наследования* и *каскадирования*. Принцип *наследования* заключается в том, что свойства CSS, объявленные для элементов-предков, наследуются элементами потомками. Но, естественно, не все свойства CSS наследуются — например, если для тега параграфа p средствами CSS задана рамка, то она не будет наследоваться ни одним тегом, содержащимся в данном теге p. Так сделано в предположении, что обрамление всех-всех вложений в тег — менее тривиальная задача, чем задание одиночной рамки. А вот если для параграфа p средствами CSS задан цвет шрифта (например, color:green;), то это свойство будет унаследовано каждым элементом-тегом, находящимся в параграфе, до тех пор, пока этому тегу не будет назначен свой цвет шрифта. Который, в свою очередь, будет теперь наследоваться всеми вложенными в него подэлементами, не распространяясь на элементы-соседи тега.

Принцип *каскадирования* применяется в случае, когда какому-то элементу [HTML](https://ru.wikipedia.org/wiki/HTML) одновременно поставлено в соответствие более одного правила CSS, то есть, когда происходит конфликт значений этих правил. Чтобы разрешить такие конфликты, вводятся правила приоритета.

* Наиболее низким приоритетом обладает стиль браузера;
* Следующим по значимости является стиль, заданный пользователем браузера в его настройках;
* И наиболее высоким приоритетом обладает стиль, заданный непосредственно автором страницы. И далее, уже в этом авторском стиле приоритеты расставляются следующим образом:
  + Самым низким приоритетом обладают стили, наследуемые в документе элементом от своих предков;
  + Более высоким приоритетом обладают стили, заданные во внешних таблицах стилей, подключённых к документу;
  + Ещё более высоким приоритетом обладают стили, заданные непосредственно селекторами всех десяти видов (см. подраздел «виды селекторов»), содержащимися в контейнерах style данного документа. Нередки случаи, когда к какому-нибудь элементу имеют отношение, задают его вид, несколько таких селекторов. Такие конфликты между ними разрешаются с помощью расчёта специфичности каждого такого селектора и применения этих селекторов к данному элементу в порядке убывания их специфичностей.

При расчёте специфичности селектора принимается во внимание:

* + - количество идентификаторов (#id) в селекторе — ((1,0,0) за каждый объявленный идентификатор в селекторе правила CSS);
    - количество классов (.class), атрибутов ([attr], [attr="value"]) и псевдоклассов (:pseudo-class) в селекторе — ((0,1,0) за каждый объявленный класс, атрибут и псевдокласс в селекторе правила CSS);
    - количество элементов (h1, input) и псевдоэлементов (::pseudo-element) в селекторе — ((0,0,1) за каждый объявленный элемент и псевдоэлемент в селекторе правила CSS).

*Принцип расчёта таков, что, например, (1,0,0) будет иметь большую специфичность, соответственно — бо́льший приоритет, чем даже (0,10,0), (0,1,0) будет иметь большую специфичность, больший приоритет, чем (0,0,10). Если же рассчитанные таким образом специфичности окажутся одинаковыми, то к элементу будет применено правило, описанное селектором, расположенным в документе ниже других*

* Ещё более высоким приоритетом обладают стили, объявленные непосредственно в теге данного элемента посредством атрибута style этого тега;
* И наконец самым высоким приоритетом обладают стили, объявленные автором страницы *или пользователем*, с помощью сопроводительного слова !important. Если таких свойств несколько, то предпочтение отдаётся в первую очередь стилям, *заданным пользователем*, а для остальных свойств (которые будут являться задаваемыми автором страницы) потребуется определить их специфичности по принципам, описанным выше, и применять эти свойства в порядке убывания этих их специфичностей.

### Пример таблицы стилей

Пример таблицы стилей (в таком виде она может быть либо размещена в отдельном файле .css либо же обрамлена тегами <style> и размещена в «шапке» той самой веб-страницы, на которую она действует):

p {

font-family: arial, helvetica, sans-serif;

}

h2 {

font-size: 20pt;

color: red;

background: white;

}

.note {

color: red;

background-color: yellow;

font-weight: bold;

}

p#paragraph1 {

padding-left: 10px;

}

a:hover {

text-decoration: none;

}

#news p {

color: blue;

}

[type="button"] {

background-color: green;

}

Здесь приведено семь правил CSS с селекторами p, h2, .note, p#paragraph1, a:hover, #news p и [type="button"].

1. Первое правило присвоено HTML-элементу **p** ([абзацу](https://ru.wikipedia.org/wiki/%D0%90%D0%B1%D0%B7%D0%B0%D1%86)) — назначен стиль. Абзацы будут отображаться шрифтом Arial или, если такой шрифт недоступен, тогда Helvetica или Sans-serif, иначе, другим шрифтом этого семейства.
2. Второе правило присвоено HTML-элементу **h2** (заголовку второго уровня). Заголовок второго уровня будет отображаться красным на белом фоне с увеличенным [размером](https://ru.wikipedia.org/wiki/%D0%9A%D0%B5%D0%B3%D0%BB%D1%8C).
3. Третье правило будет применено к *любому* элементу, атрибут **class** которого равен «note». Например, к параграфу: <p class="note">Этот абзац будет выведен полужирным шрифтом красного цвета на жёлтом фоне. </p>
4. Четвёртое правило будет применяться только к элементу p, атрибут id которого равен paragraph1. Такой элемент будет иметь внутренний отступ в 10 пикселей (padding).
5. Пятое правило определяет стиль *hover* для элементов **a** — [гиперссылок](https://ru.wikipedia.org/wiki/%D0%93%D0%B8%D0%BF%D0%B5%D1%80%D1%81%D1%81%D1%8B%D0%BB%D0%BA%D0%B0). [По умолчанию](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE_%D1%83%D0%BC%D0%BE%D0%BB%D1%87%D0%B0%D0%BD%D0%B8%D1%8E), в большинстве браузеров текст элементов **a** подчёркивается. Это правило уберёт подчёркивание, когда указатель мыши находится над этими элементами.
6. Шестое правило, применяется для элементов **p**, которые находятся внутри *какого-либо* элемента с атрибутом id, равным «news» (#news p — это типичный случай селектора потомков, см. [5-й пункт списка выше](https://ru.wikipedia.org/wiki/CSS#Виды_селекторов)).
7. Седьмое правило применяется для всех элементов, у которых атрибут type равен **button**. Например, это правило будет применено к элементу <input type="button"> (обычная кнопка), изменив его цвет фона на зелёный.

## CSS-вёрстка

До появления CSS оформление веб-страниц осуществлялось исключительно средствами [HTML](https://ru.wikipedia.org/wiki/HTML), непосредственно внутри содержимого документа. Однако с появлением CSS стало возможным принципиальное разделение содержания и представления документа. За счёт этого нововведения стало возможным лёгкое применение единого стиля оформления для массы схожих документов, а также быстрое изменение этого оформления.

Преимущества:

* Несколько дизайнов страницы для разных устройств просмотра. Например, на экране дизайн будет рассчитан на большую ширину, во время печати меню не будет выводиться, а на [КПК](https://ru.wikipedia.org/wiki/%D0%9A%D0%B0%D1%80%D0%BC%D0%B0%D0%BD%D0%BD%D1%8B%D0%B9_%D0%BF%D0%B5%D1%80%D1%81%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80) и сотовом телефоне меню будет следовать за содержимым.
* Уменьшение времени загрузки страниц сайта за счет переноса правил представления данных в отдельный CSS-файл. В этом случае браузер загружает только структуру документа и данные, хранимые на странице, а представление этих данных загружается браузером только один раз и может быть [закэшировано](https://ru.wikipedia.org/wiki/%D0%9A%D1%8D%D1%88).
* Простота последующего изменения дизайна. Не нужно править каждую страницу, а достаточно лишь изменить CSS-файл.
* Дополнительные возможности оформления. Например, с помощью CSS-вёрстки можно сделать блок текста, который остальной текст будет обтекать (например для меню) или сделать так, чтобы меню было всегда видно при прокрутке страницы.

Недостатки:

* Различное отображение вёрстки в различных браузерах (особенно устаревших), которые по-разному интерпретируют одни и те же данные CSS.
* Часто встречающаяся необходимость на практике исправлять не только один CSS-файл, но и теги HTML, которые сложным и ненаглядным способом связаны с селекторами CSS, что иногда сводит на нет простоту применения единых файлов стилей и значительно увеличивает время редактирования и тестирования.

# **Переменные в CSS**

Если вы разработчик, то вы точно хорошо знакомы с переменными, и возможно, они одни из ваших лучших друзей. По определению, переменная — это временное хранилище, которое содержит некое значение величины или информации.  
Вышел первый релиз рабочего черновика [CSS Переменных (CSS Variables)](http://www.w3.org/TR/css-variables/).  
  


### Почему CSS переменные?

Переменные в CSS — эта та штука, о которой разработчики спрашивали и хотели довольно [долгое время](http://disruptive-innovations.com/zoo/cssvariables/).   
Подумайте обо всех этих цветах (colors), высотах (heights), ширинах (widths) и размерах (sizes): как бы было прекрасно объявить их всего лишь один раз. И наконец, пришло время того, чего мы так долго ждали: **писать меньше, но делать больше**.

### Установившиеся практики в CSS

Когда люди просят об объявлении переменных цвета в css (color), добавление комментариев в верней части CSS-файла было чем-то вроде симуляции поведения переменных:

/\*--------------------------

link color: #99D1FF (light blue)

box color: #555 (dark gray)

--------------------------\*/

Позже, чтобы обновить значения, приходилось делать поиск и замену.

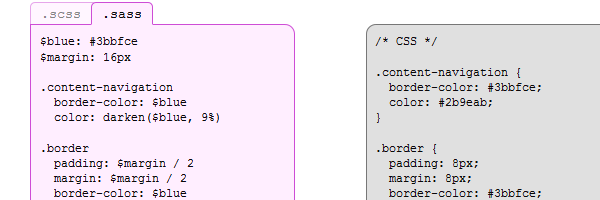
### Как это делается в LESS/Sass

Идея использовать переменные для таблицы стилей было одной из тех причин, по которым появились [LESS](http://lesscss.org/) и [Sass](http://sass-lang.com/).

#### LESS



#### Sass



### Как это будет работать теперь

Прежде всего, не забывайте, что это ни один из браузеров пока не поддерживает. Но это то, как оно будет работать в будущем:   
**var-foo** для определения, **var(foo)** для использования.  
Следуя черновикам:

Любое имя свойства, начинающееся с префикса “var-” является свойством переменной. (Any property name starting with the prefix “var-” is a variable property)

#### Пример

Следующее правило декларирует имя свойства “var-header-color” для элемента root и присваивает для него значение “#99D1FF”:

:root {

var-header-color: #99D1FF;

}

Далее, его значение может передаваться с помощью переменной “header-color”:

h1 {

color: var(header-color);

}

Использование переменных цвета в определении градиентов также может быть очень полезным. Вам всего лишь нужно будет заменить значение переменных, и вуаля: все градиенты обновились. Звучит довольно круто, как по мне.  
Также, при создании макета, применив переменные и функцию [calc()](http://www.w3.org/TR/css3-values/#calc0) можно сделать интересные вычисления.

**Основные свойства CSS, используемые для управления стилями элементов HTML документов.**

**Свойства CSS**

CSS позволяет манипулировать следующими свойствами элементов:

**Свойства шрифта**

**font-family** - определяет используемый элементом шрифт. Если указать URL(file), то шрифт автоматически установится на компьютер пользователя;   
**font-style** - стиль шрифта (normal, italic);   
**font-variant** - варианты отображения шрифта (normal, small-caps);   
**font-weight** - жирность шрифта (normal, bold, bolder, lighter, значение от 100 до 900);   
**font-size** - размер шрифта (размер, xx-small, x-small, small, medium, large, x-large, xx-large, smaller, larger);   
**font** - обобщает вышеперечисленные свойства (любая комбинация из вышеперечисленных значений);

**Свойства текста**

**word-spacing** - расстояние между словами (значение, normal);   
**text-decoration** - декорация текста (none, underline, overline, line-through, blink);   
**letter-spacing** - расстояние между буквами (значение, normal);   
**vertical-align** - позиционирование по отношению к другим элементам стоящим в одном ряду (baseline, sub, super, top-text, top, middle, bottom, bottom-text, %);   
**text-transform** - изменение текста (none, Capitalize, UPPERCASE, lowercase);   
**text-align** - положение текста (left, right, center, justify);   
**text-indent** - отступ (значение, %);   
**line-height** - отступ сверху (normal, значение, %);

**Свойства фон и цвет**

**color** - цвет элемента (значение);   
**backgroung-color** - цвет фона элемента (значение);   
**background-image** - изображение фон (none, URL);   
**background-repeat** - варианты повторения фонового изображения (repeat, repeat-x, repeat-y, no-repeat);   
**background-attachment** - возможность прокрутки фонового изображения (scroll, fixed);   
**background-position** - положение фонового изображения (%ширины%высоты, top, middle, bottom, left, center, right);   
**background** - обобщает вышеперечисленные свойства (любая комбинация из вышеперечисленных значений);

**Свойства блока**

**margin-top** - определяет отступ сверху (значение, %, auto);   
**margin-right** - определяет отступ справа (значение, %, auto);   
**margin-bottom** - определяет отступ снизу (значение, %, auto);   
**margin-left** - определяет отступ слева (значение, %, auto);   
**margin** - обобщает все вышеперечисленные свойства;   
**padding-top** - отступ от верхнего border'а (значение, %);   
**padding-right** - отступ от правого border'а (значение, %);   
**padding-bottom** - отступ от нижнего border'а (значение, %);   
**padding-left** - отступ от левого border'а (значение, %);   
**padding** - обобщает все вышеперечисленные свойства;   
**border-top-width** - толщина верхнего border'а (значение, thin, medium, thick);   
**border-right-width** - толщина правого border'а (значение, thin, medium, thick);   
**border-bottom-width** - толщина нижнего border'а (значение, thin, medium, thick);   
**border-left-width** - толщина левого border'а (значение, thin, medium, thick);   
**border-width** - обобщает все вышеперечисленные свойства;   
**border-color** - Цвет border'а. (значение);   
**border-style** - стиль border'ов (none, dotted, dashed, solid, double, groove, ridge, inset, outset);   
**border-top** - обобщает вышеперечисленные свойства для верхнего border'а;   
**border-right** -обобщает вышеперечисленные свойства для правого border'а;   
**border-left** - обобщает вышеперечисленные свойства для левого border'а;   
**border-bottom** - обобщает вышеперечисленные свойства для нижнего border'а;   
**border** - обобщает все вышеперечисленные свойства;   
**width** - ширина элемента (значение, %);   
**height** - высота элемента (значение, %);   
**float** - расположение элемента (left, right, none);   
**clear** - расположение других элементов вокруг данного (left, right, both, none);

**Классификационные свойства**

**display** - определяет, как будет отображаться элемент (none, block, inline, list-item);   
**white-space** - определяет, как будут отображаться пробелы между элементами (normal, pre, nowrap);   
**list-style-type** - определяет вид list-item маркера в списках (disc, circle, square, decimal, lower-roman, upper-roman, lower-alpha, upper-alpha, none);   
**list-style-image** - задает вид list-item маркера из картинки (none, URL);   
**list-style-position** - определяет положение маркера в зависимости от list item элемента (inside, outside);   
**list-style** - обобщает вышеперечисленные свойства;

**Свойства элемента**

**position** - определяет, как будет отображаться элемент по отношению к другим элементам документа (relative, absolute);   
**top** - определяет позицию элемента TOP относительно элемента родителя (значение, %);   
**left** - определяет позицию элемента LEFT относительно элемента родителя (значение,%);   
**width** - определяет ширину элемента (значение, %, auto);   
**height** - определяет высоту элемента (значение, %, auto);   
**overflow** - режим отображения содержимого элемента, при несоответствии размера элемента, размеру содержимого (non, clip, scroll);   
**visibility** - управление видимостью элемента в документе (hidden, " ");

**Задание:**

Подготовить макет страницы с использованием css, выбрать любой макет из представленных, содержание определить самостоятельно.

