CSS



Хокон Виум Ли род. 27 июля 1965, Халден, Норвегия - учёный, специалист в области информатики в 1994 г. предложивший каскадные таблицы стилей (CSS).

CSS — одна из широкого спектра технологий, одобренных консорциумом W3C и получивших общее название «стандарты Web». В 1990-х годах стала ясна необходимость стандартизировать Web, создать какие-то единые правила, по которым программисты и веб-дизайнеры проектировали бы сайты. Так появились языки HTML 4.01 и XHTML, и стандарт CSS.

В начале 1990-х различные браузеры имели свои стили для отображения веб-страниц. HTML развивался очень быстро и был способен удовлетворить все существовавшие на тот момент потребности по оформлению информации, поэтому CSS не получил тогда широкого признания.

Термин «каскадные таблицы стилей» был предложен Хоконом Ли в 1994 году. Совместно с Бертом Босом он стал развивать CSS.

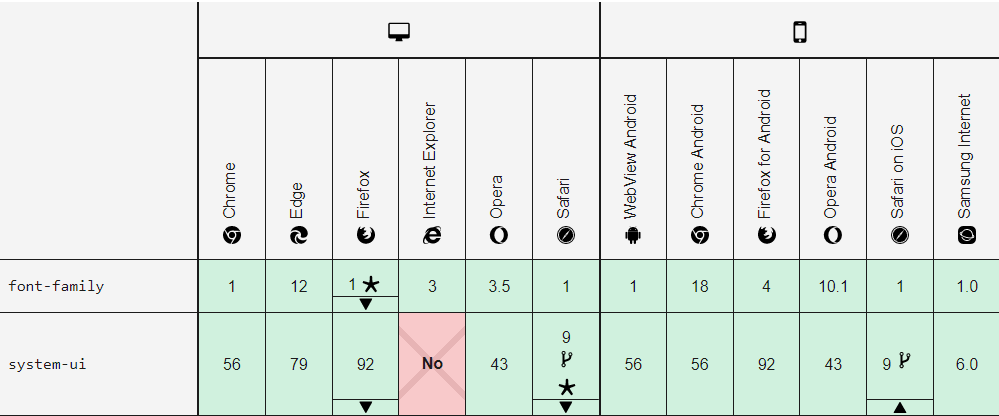
В отличие от многих существовавших на тот момент языков стиля, CSS использует наследование от родителя к потомку, поэтому разработчик может определить разные стили, основываясь на уже определённых ранее стилях.

В середине 1990-х Консорциум Всемирной паутины (W3C) стал проявлять интерес к CSS, и в декабре 1996 года была издана рекомендация CSS1

Война браузеров

<https://ru.wikipedia.org/wiki/%D0%92%D0%BE%D0%B9%D0%BD%D0%B0_%D0%B1%D1%80%D0%B0%D1%83%D0%B7%D0%B5%D1%80%D0%BE%D0%B2>

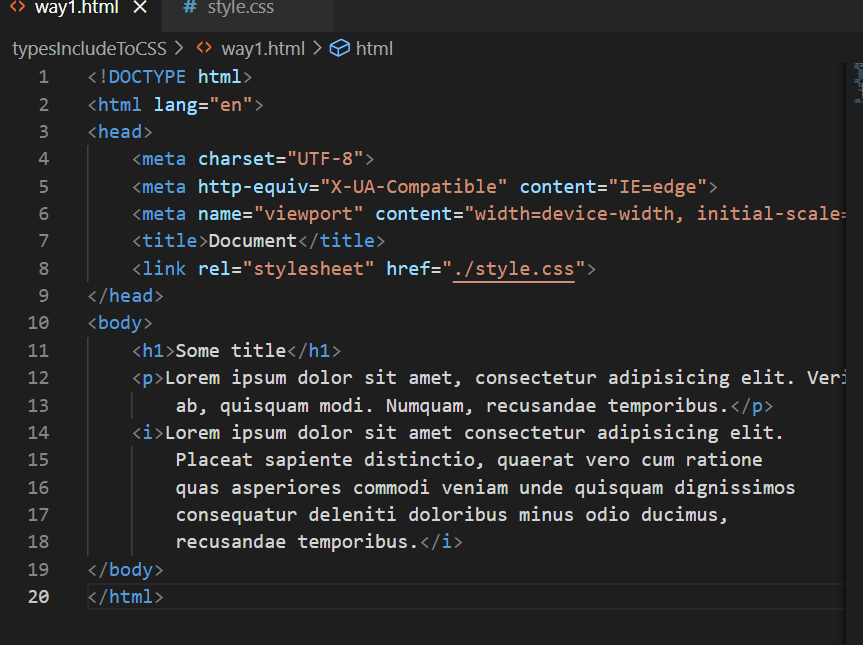
<https://developer.mozilla.org/ru/docs/Learn/CSS/First_steps/What_is_CSS>



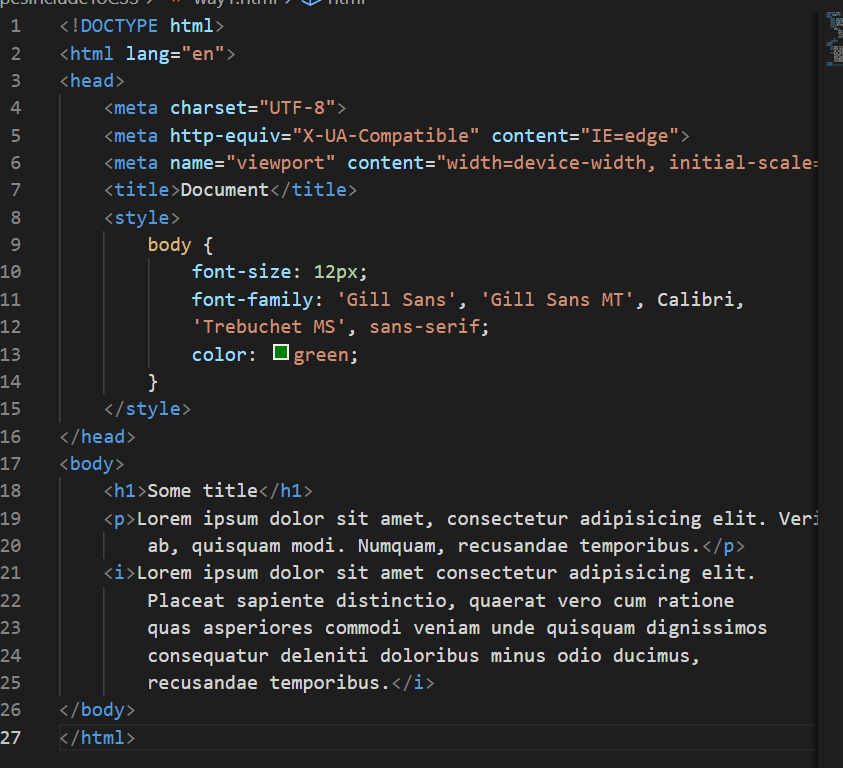
Let’s start!

# Способов добавления css на страницу html:

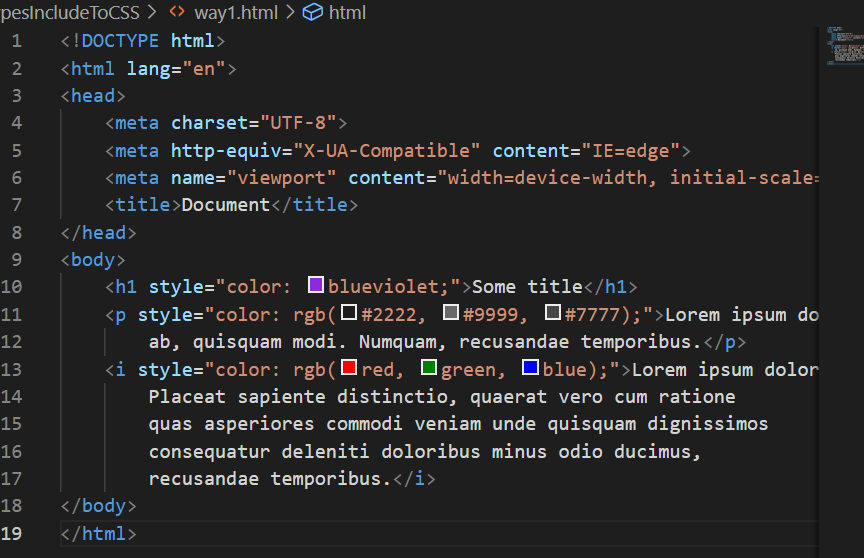
# 1. Подключить файл стиля



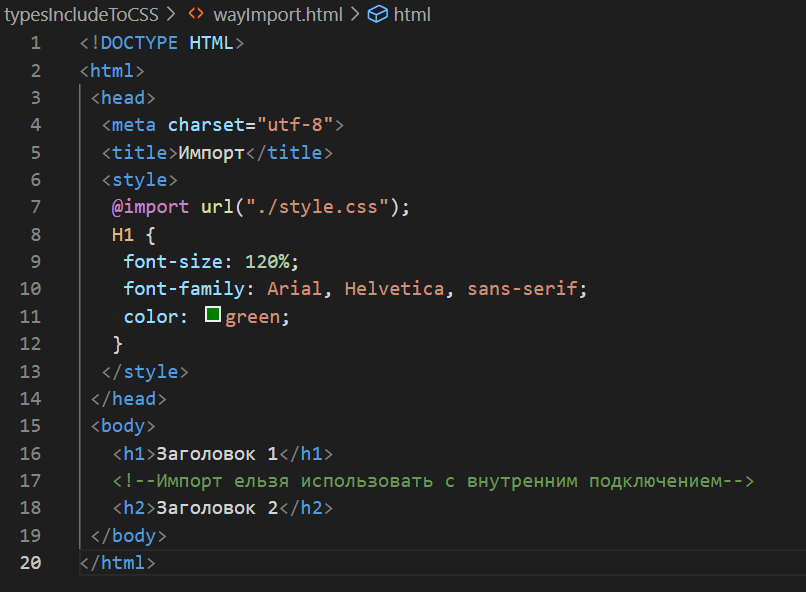
2. Глобальный стиль



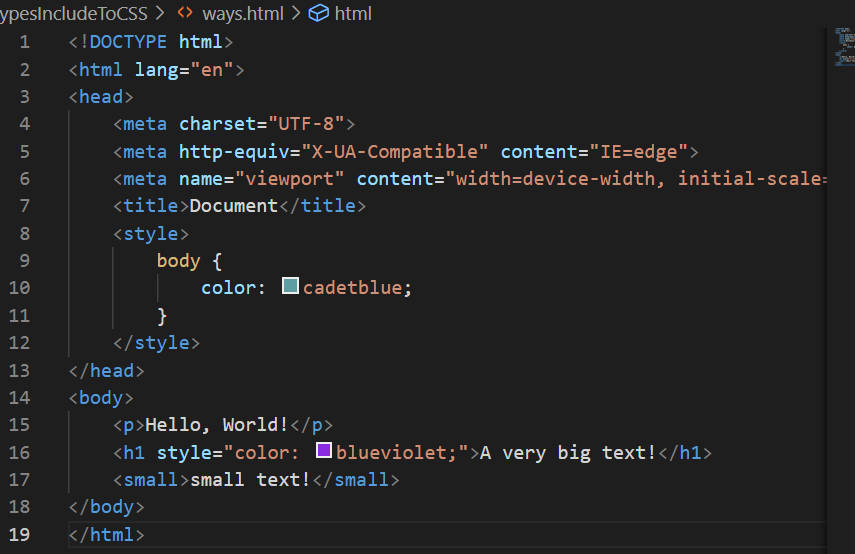
3. Внутренний стиль

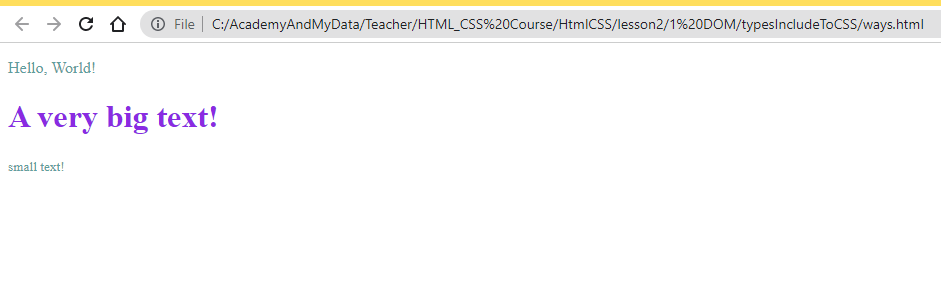


4. Импорт CSS



5. Смешанный вид подключения

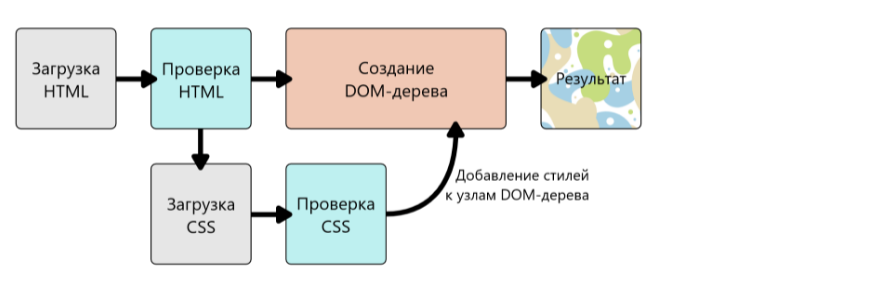




**Как работает CSS**

Когда браузер отображает документ, он должен совместить его содержимое с его стилями. Этот процесс идёт в несколько этапов, о которых мы сейчас поговорим. Держите в уме, что это очень упрощённая версия того как браузер действительно загружает веб-страницу, а также то, что разные браузеры делают это по разному. Но, происходит, грубо говоря, следующее:

1. Браузер получает HTML-страницу (например, из Интернета)
2. Преобразует [HTML](https://developer.mozilla.org/ru/docs/Glossary/HTML) в [DOM](https://developer.mozilla.org/ru/docs/Glossary/DOM) (Document Object Model). DOM (или DOM-дерево) - это представление страницы в памяти компьютера. Подробнее на DOM мы остановимся чуть позже.
3. Затем браузер забирает все ресурсы и описания, связанные с HTML-документом, например: встроенные картинки, видео ... и  стили CSS! JavaScript присоединяется чуть позже и мы пока не будем говорить об этом, чтобы все не усложнять.
4. После этого браузер анализирует полученный CSS код, сортирует описанные там правила в зависимости от их селекторов и раскладывает их в различные «корзины»: элементы, классы, идентификаторы(ID) и т.п. Основываясь на найденных селекторах браузер понимает какие правила относятся к определённым «узлам» в DOM-дереве и применяет их по мере необходимости (этот промежуточный шаг называют «формированием дерева представления» или «формированием дерева рендеринга»)
5. Дерево представления (render tree) формируется в том порядке, в каком оно затем должно будет отображаться, когда все правила будут применены.
6. Затем происходит визуальное отображение контента на странице (этот этап называется «отрисовкой»)

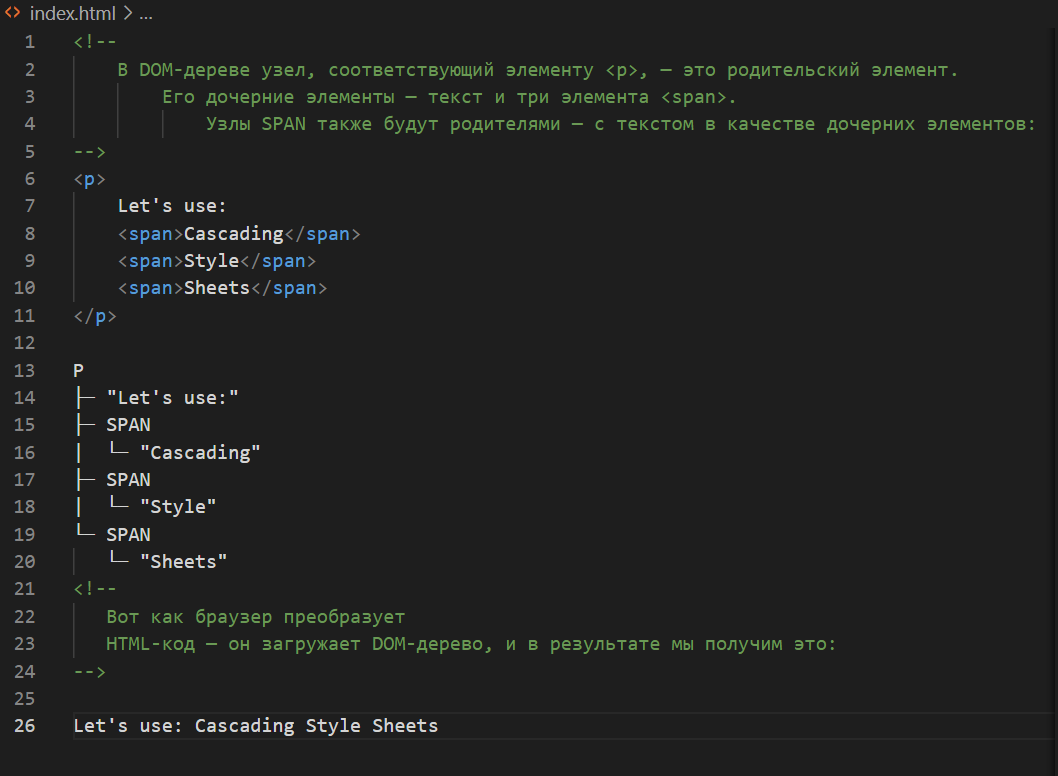


## 

## [DOM-дерево](https://developer.mozilla.org/ru/docs/Learn/CSS/First_steps/How_CSS_works#dom-%D0%B4%D0%B5%D1%80%D0%B5%D0%B2%D0%BE)

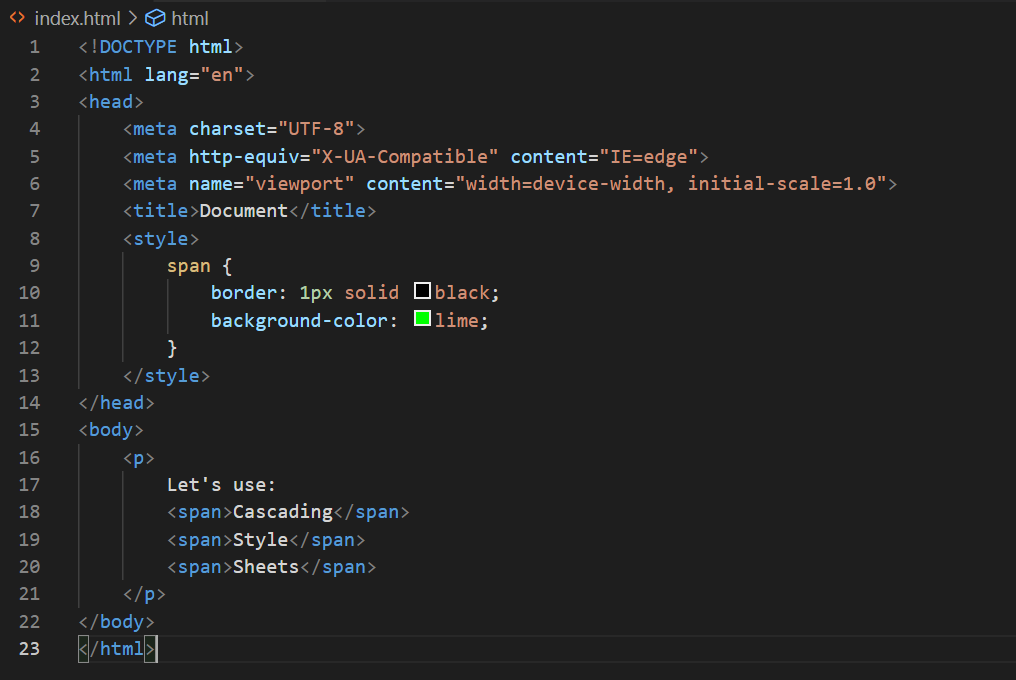
DOM напоминает дерево. Каждый элемент, атрибут, отрывок текста становится DOM node в разметке. DOM-узлы определяются их отношением с другими узлами. Некоторые родительские элементы имеют дочерние, а у дочерних элементов есть братские.

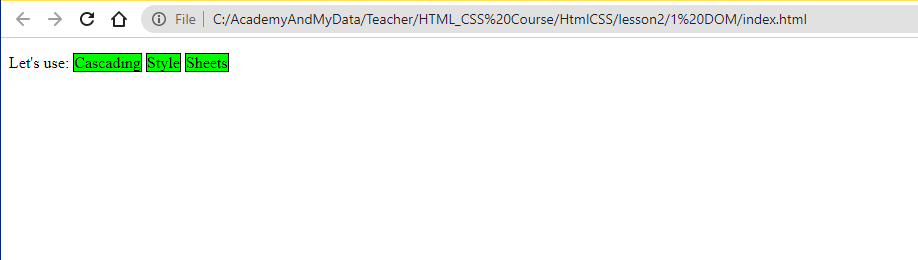
Понимание DOM позволит вам разрабатывать, отлаживать и поддерживать ваш CSS, потому что именно в DOM-дереве ваши таблицы стилей и код встречаются.



Допустим, мы добавили таблицу стилей к нашему примеру:

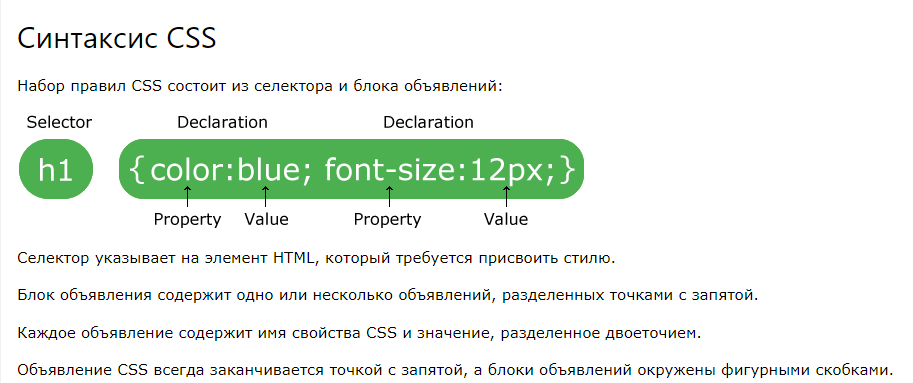
Браузер загрузит HTML-код, преобразует его в DOM и только потом загрузит CSS. Так как у нас всего одно правило,  браузер загрузит CSS очень быстро! Это правило будет добавлено к каждому из трёх элементов <span>. После этого информация будет отображена на экране.





В соответствии с объектной моделью документа («Document Object Model», коротко DOM), каждый HTML-тег является объектом. Вложенные теги являются «детьми» родительского элемента. Текст, который находится внутри тега, также является объектом.

Синтаксис:



**Селектор**определяет, к какому элементу применять то или иное CSS-правило.

Базовые селекторы:

Универсальный селектор

Выбирает все элементы.

Синтаксис: \*

Пример: \* будет соответствовать всем элементам на странице.

Селекторы по типу элемента

Этот базовый селектор выбирает тип элементов, к которым будет применяться правило.

Синтаксис: элемент

Пример: селектор input выберет все элементы <input>.

Селекторы по классу

Этот базовый селектор выбирает элементы, основываясь на значении их атрибута class.

Синтаксис: .имяКласса

Пример: селектор .index выберет все элементы с соответствующим классом (который был определён в атрибуте class="index").

Селекторы по идентификатору

Этот базовый селектор выбирает элементы, основываясь на значении их id атрибута. Не забывайте, что идентификатор должен быть уникальным, т. е. использоваться только для одного элемента в HTML-документе.

Синтаксис: #имяИдентификатора

Пример: селектор #toc выберет элемент с идентификатором toc (который был определён в атрибуте id="toc").

Селекторы по атрибуту

Этот селектор выбирает все элементы, имеющие данный атрибут или атрибут с определённым значением.

Синтаксис: [attr] [attr=value] [attr~=value] [attr|=value] [attr^=value] [attr$=value] [attr\*=value]

Пример: селектор [autoplay] выберет все элементы, у которых есть атрибут autoplay (независимо от его значения).

Ещё пример: a[href$=".jpg"] выберет все ссылки, у которых адрес заканчивается на ".jpg".

Ещё пример: a[href^="https"] выберет все ссылки, у которых адрес начинается на "https".

Комбинаторы

Комбинатор запятая

Комбинатор , это способ группировки, он выбирает все совпадающие узлы.

Синтаксис: A, B

Пример: div, span выберет оба элемента - и <div> и <span>.

Комбинатор потомков

Комбинатор ' ' (пробел) выбирает элементы, которые находятся внутри указанного элемента (вне зависимости от уровня вложенности).

Синтаксис: A B

Пример: селектор div span выберет все элементы <span>, которые находятся внутри элемента <div>.

Дочерние селекторы

Комбинатор '>' в отличие от пробела выбирает только те элементы, которые являются дочерними непосредственно по отношению к указанному элементу.

Синтаксис: A > B

Пример: селектор ul > li выберет только дочерние элементы <li>, которые находятся внутри, на первом уровне вложенности по отношению к элементу <ul>.

Комбинатор всех соседних элементов

Комбинатор '~' выбирает элементы, которые находятся на этом же уровне вложенности, после указанного элемента, с тем же родителем.

Синтаксис: A ~ B

Пример: p ~ span выберет все элементы <span>, которые находятся после элемента <p> внутри одного родителя.

Комбинатор следующего соседнего элемента

Комбинатор '+' выбирает элемент, который находится непосредственно после указанного элемента, если у них общий родитель.

Синтаксис: A + B

Пример: селектор ul + li выберет любой <li> элемент, который находится непосредственно после элемента <ul>.

Псевдо

Псевдоклассы

Знак : позволяет выбрать элементы, основываясь на информации, которой нет в дереве элементов.

Пример: a:visited соответствует всем элементам <a> которые имеют статус "посещённые".

Ещё пример: div:hover соответствует элементу, над которым проходит указатель мыши.

Ещё пример: input:focus соответствует полю ввода, которое получило фокус.

Псевдоэлементы

Знак :: позволяет выбрать вещи, которых нет в HTML.

Пример: p::first-line соответствует первой линии абзаца <p>.

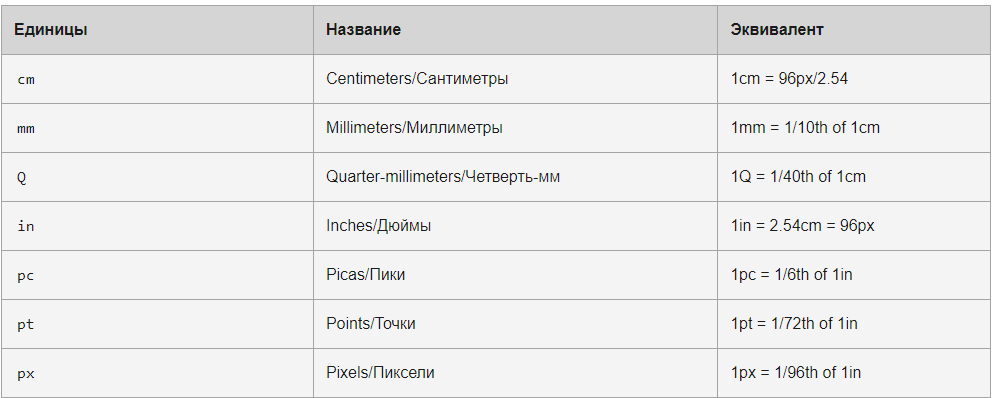
Единицы измерения CSS

### [Длины](https://developer.mozilla.org/ru/docs/Learn/CSS/Building_blocks/Values_and_units#%D0%B4%D0%BB%D0%B8%D0%BD%D1%8B):

Числовой тип, с которым вы будете сталкиваться чаще всего это <length> (длина), например 10px (пиксели) или 30em. Существует два типа длин используемых в CSS — относительные и абсолютные. Важно знать разницу для того, чтобы понимать, насколько большими станут вещи.

#### Абсолютные единицы длины:

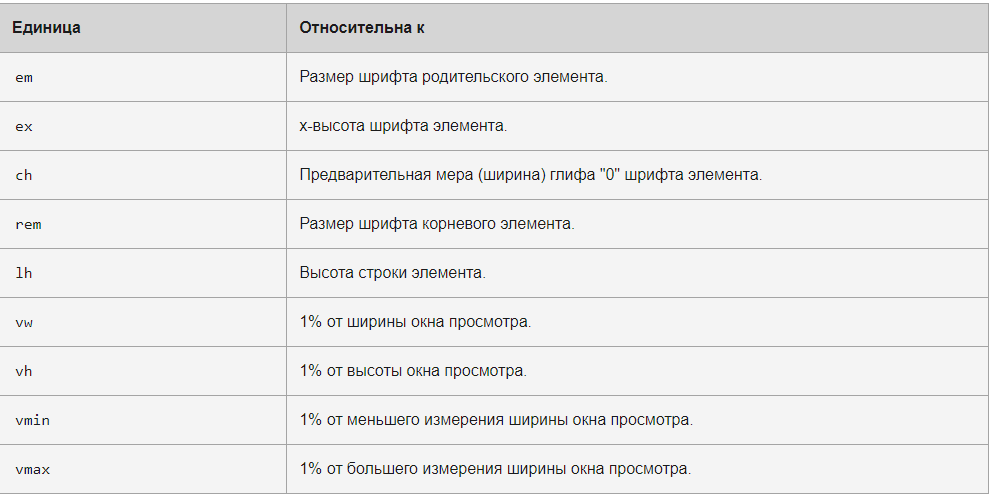
Ниже приведены все единицы абсолютной длины — они не являются относительными к чему-либо и обычно считаются всегда одинакового размера.

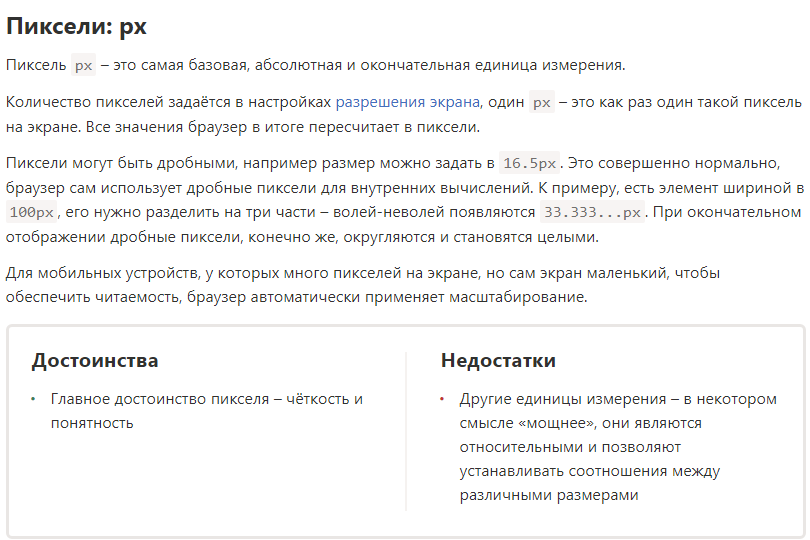


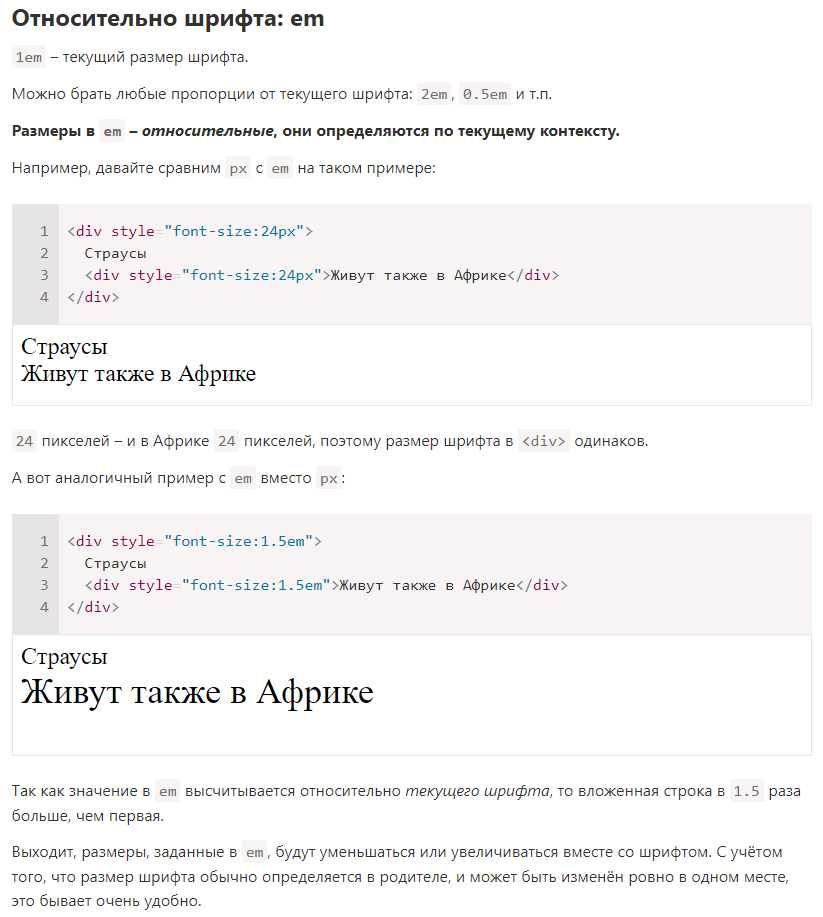
Большинство из этих значений больше полезны при использовании печати, чем для вывода на экран. Например, мы обычно не используем cm (сантиметры) на экране. Единственное значение которое вы в основном будете использовать это px (пиксели).

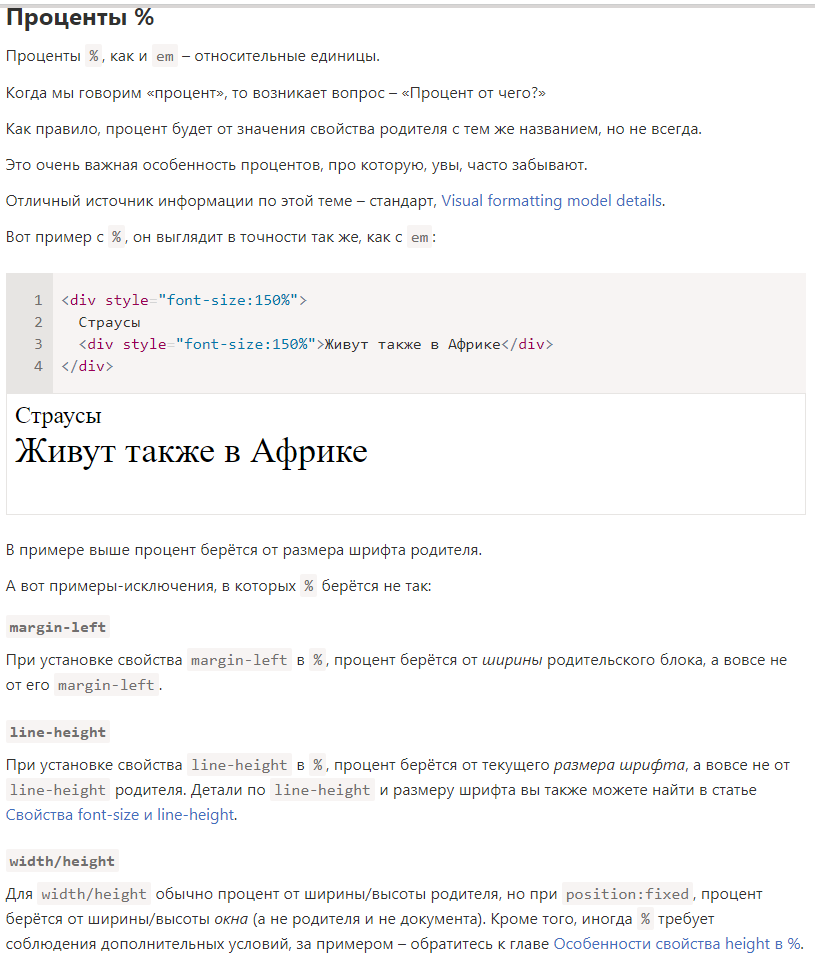
#### Единицы относительной длины

Относительные единицы длин являются относительными к чему-то ещё, возможно к размеру родительского шрифта или к размеру окна просмотра. Преимущество использования относительных единиц состоит в том, что при тщательном планировании вы можете сделать так, чтобы размер текста или других элементов масштабировался относительно всего остального на странице. Некоторые наиболее используемые единицы веб-разработки перечислены в таблице ниже.





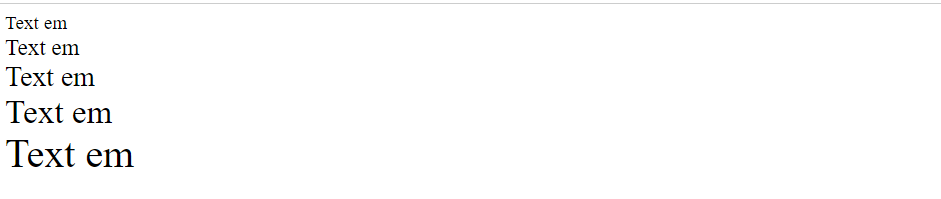




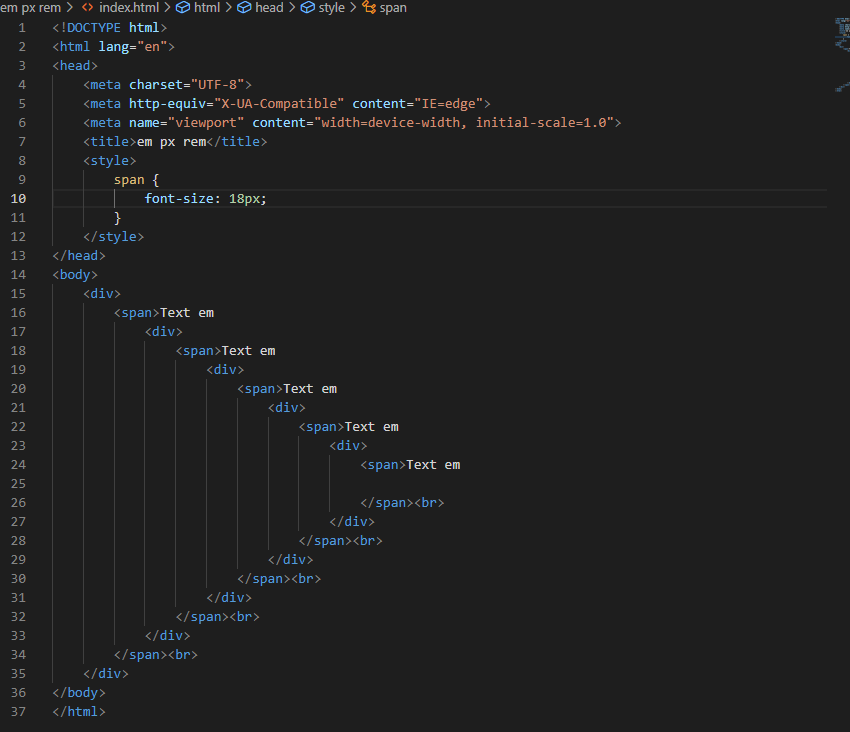
Единица rem: смесь px и em

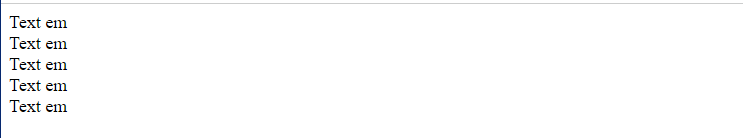
em





Px



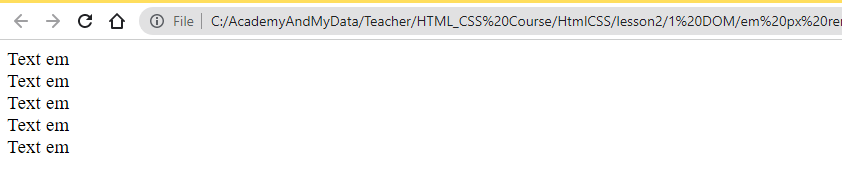


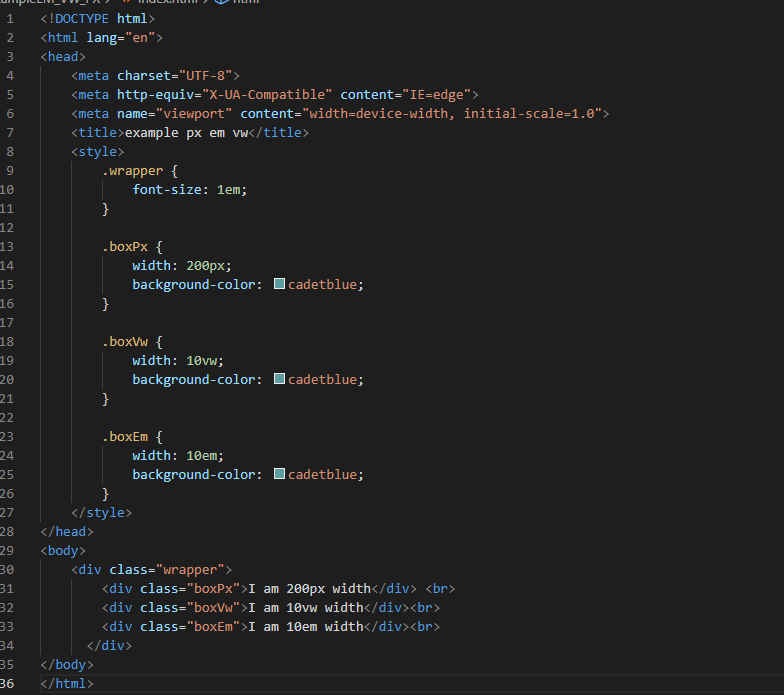
Rem

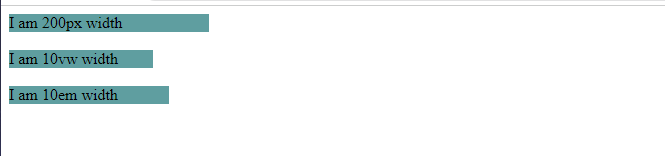
Единица rem задаёт размер относительно размера шрифта элемента <html>.

Как правило, браузеры ставят этому элементу некоторый разумный размер по умолчанию, который мы, конечно, можем переопределить и использовать rem для задания шрифтов внутри относительно него:

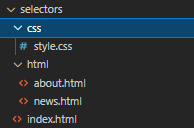




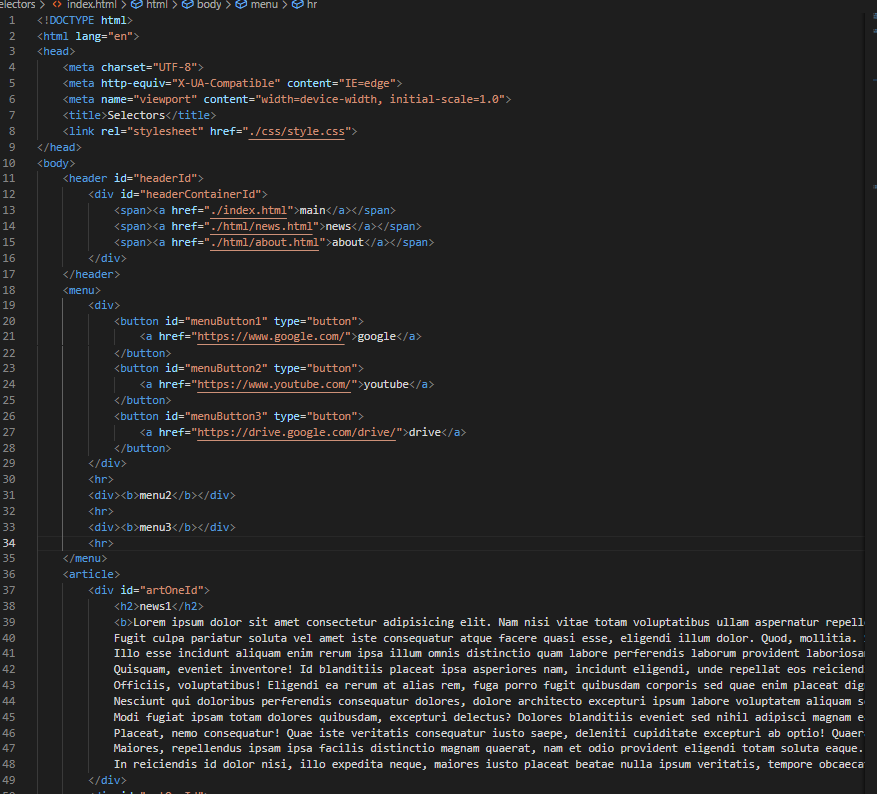


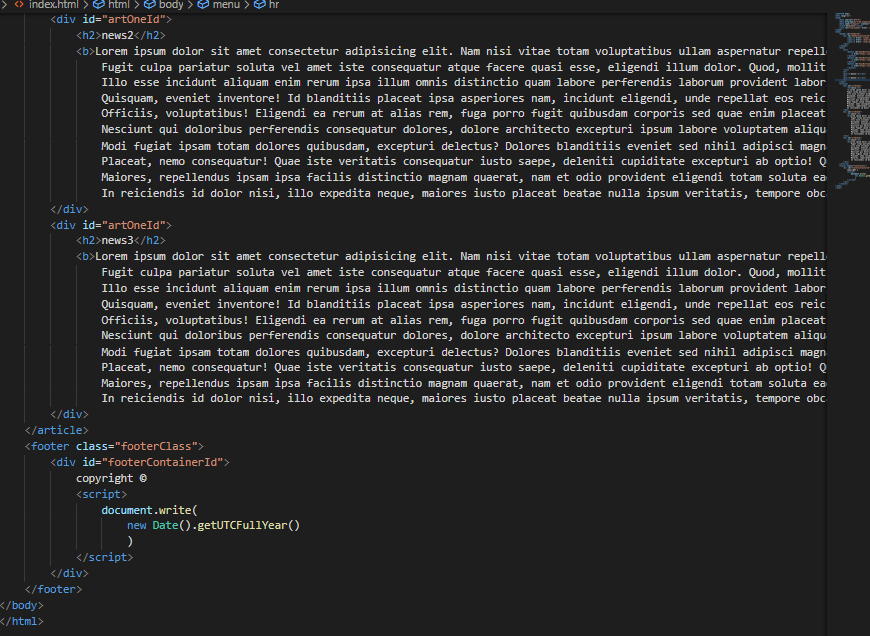


Структура проекта:



Index.html

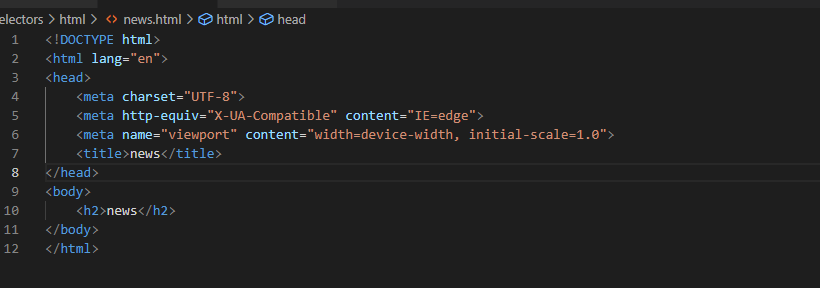




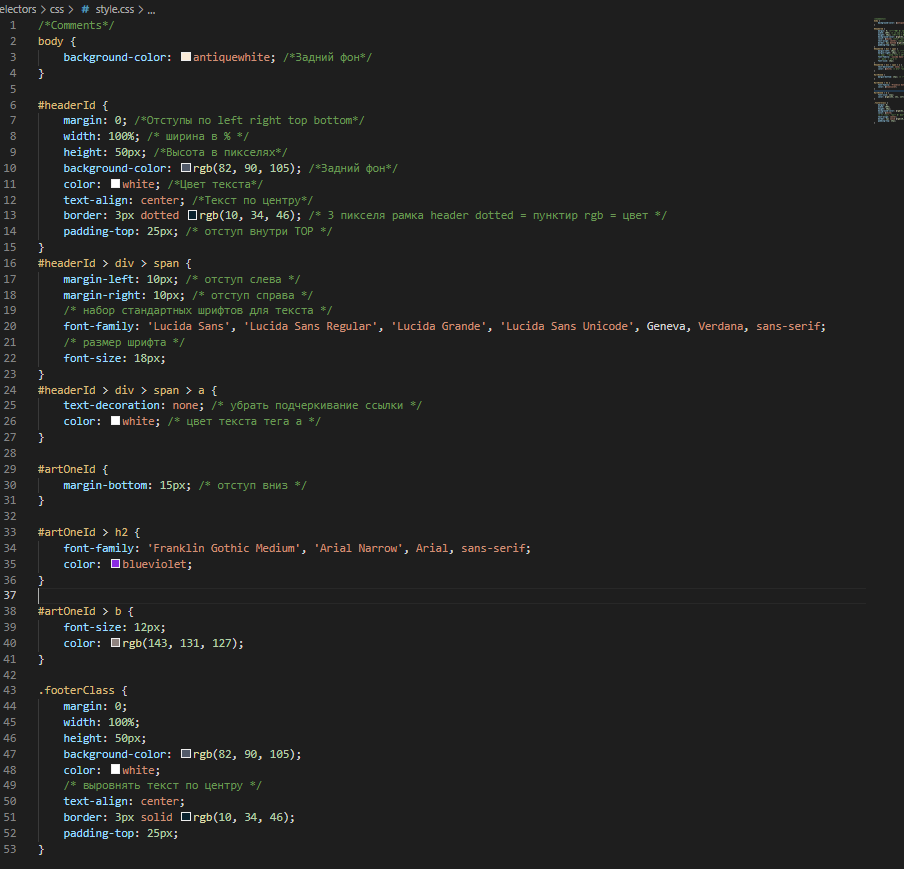
About.html



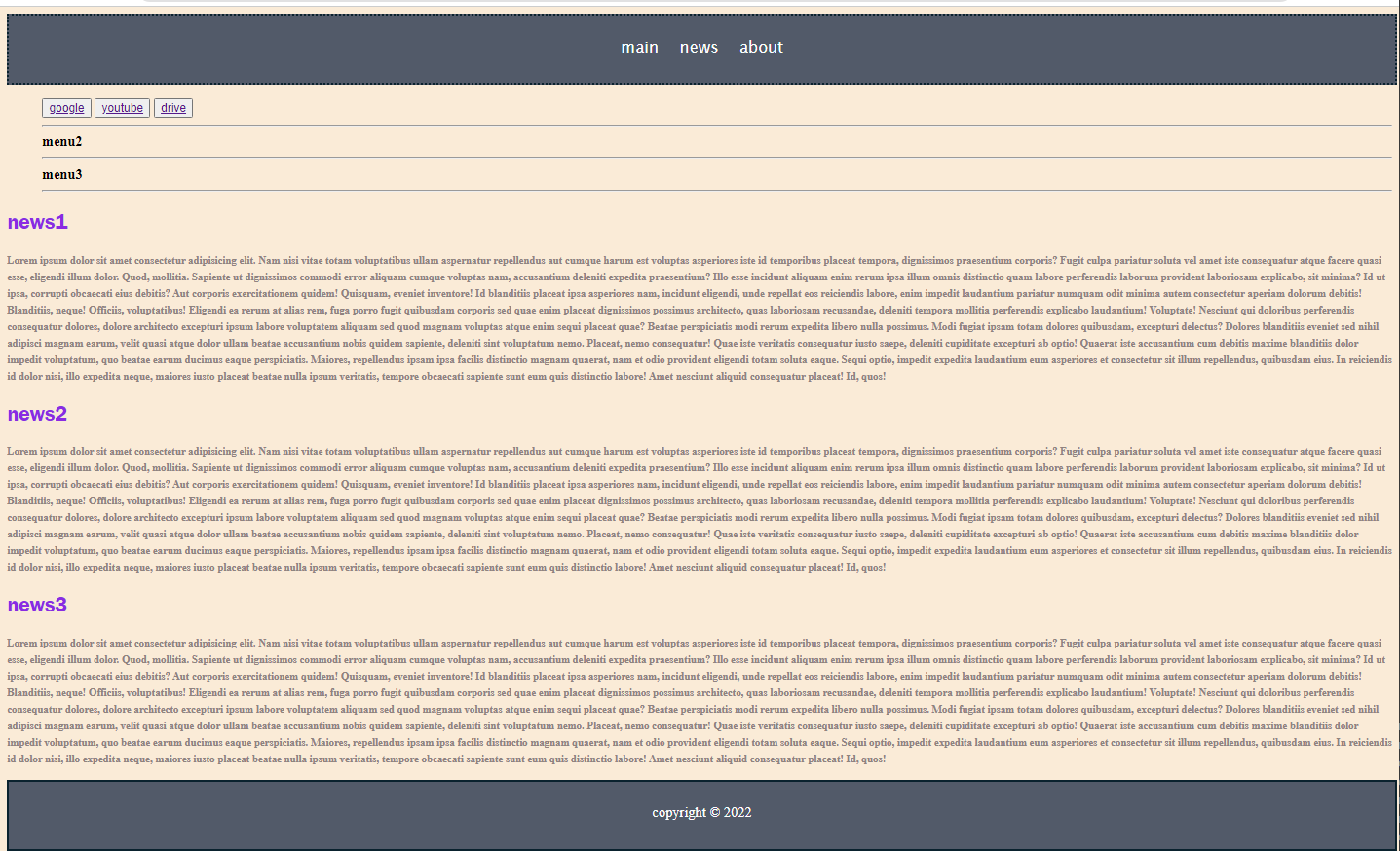
News.html



Style.css



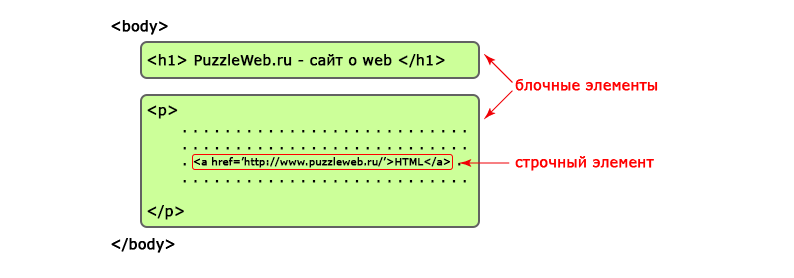
Chrome



Блочные теги:

представляют собой большие строительные блоки вашей веб-страницы. При отображении браузер автоматически добавляет разрыв строки до и после блочного элемента, при этом он занимает всю доступную ширину (по умолчанию отображается на веб-странице в виде прямоугольника)

Обычно блочные элементы используются, чтобы разделить содержимое веб-страницы на логические блоки (например: верхний колонтитул (шапка сайта), меню, блок с контентом, нижний колонтитул (footer) и др.).



## Строчные теги:

|  |  |
| --- | --- |
| **Тег** | **Краткое описание** |
| <a> | Создает гиперссылку. |
| <abbr> | Определяет текст как аббревиатуру. |
| <audio> | Определяет звуковое содержимое. |
| <b> | Выделяет текст жирным шрифтом. |
| <bdo> | Определяет направление отображения текста. |
| <button> | Создает кнопку. |
| <canvas> | Определяет область для рисования графики. |
| <cite> | Определяет заголовок для работы - преобразует текст в курсивный. |
| <code> | Определяет кусок программного кода - преобразует текст в моноширинный. |
| <del> | Определяет текст, который был удален из документа - отображается перечеркнутым текстом. |
| <dfn> | Выделяет определения термина - преобразует шрифт в наклонный. |
| <em> | Определяет выделенный текст - преобразует текст в курсивный. |
| <i> | Преобразует текст в курсивный. |
| <iframe> | Определяет встроенный frame. |
| <img> | Определяет изображение. |
| <input> | Создаeт поле для ввода данных. |
| <ins> | Определяет текст, который был добавлен в документ - отображает текст подчеркнутым. |
| <kbd> | Определяет текст вводимый с клавиатуры - преобразует текст в моноширинный. |
| <mark> | Определяет важную часть текста. |
| <meter> | Определяет скалярное измерение в пределах известного диапазона. |
| <q> | Определяет короткую цитату. |
| <rp> | Определяет, что показывать браузеру, который не поддерживает тег <ruby>. |
| <rt> | Добавляет аннотацию сверху или снизу от символов, заключенных в элементе <ruby>. |
| <s> | Определяет текст, который больше не является правильным - отображает текст перечеркнутым. |
| <samp> | Определяет текст, который является результатом вывода компьютерной программы. |
| <small> | Определяет текст маленького размера. |
| <select> | Создает выпадающий список. |
| <span> | Определяет строчный элемент документа. |
| <strong> | Определяет важный текст - преобразует шрифт в полужирный. |
| <sub> | Определяет текст в нижнем индексе. |
| <sup> | Определяет текст в верхнем индексе. |
| <td> | Создает ячейку таблицы. |
| <textarea> | Создает многострочное текстовое поле. |
| <th> | Создает заголовочную ячейку в таблице. |
| <var> | Определяет переменную - выделяет текст курсивом. |
| <video> | Добавляет на страницу видео файл. |

## Блочные теги:

|  |  |
| --- | --- |
| **Тег** | **Краткое описание** |
| <address> | Определяет контактную информацию автора документа/статьи. |
| <artical> | Определяет текст как статью, новость и др. |
| <aside> | Определяет контент в стороне от содержимого страницы. |
| <blockquote> | Выделяет текст с другого источника, как блочную цитату. |
| <dd> | Cоздает описание элемента в списке определений. |
| <div> | Определяет раздел документа. |
| <dl> | Создает список определений. |
| <dt> | Oпределяет термин в списке определений. |
| <figure> | Группирует элементы страницы. |
| <footer> | Нижняя часть документа. |
| <form> | Определяет HTML форму. |
| <h1> - <h6> | Определяют HTML заголовки. |
| <header> | Задает "шапку" сайта или раздела. |
| <hr> | Создает горизонтальную линию. |
| <li> | Определяет элемент списка. |
| <nav> | Определяет группу ссылок для навигации. |
| <ol> | Создает нумерованный(упорядоченный) список. |
| <p> | Определяет абзац. |
| <pre> | Оставляет содержимое в первоначальном виде. |
| <ruby> | Определяет небольшую аннотацию (для типографии Восточной Азии). |
| <section> | Определяет разделы документа. |
| <table> | Создает таблицу. |
| <tr> | Создает строку в таблице. |
| <ul> | Определяет маркированный(неупорядоченный) список. |

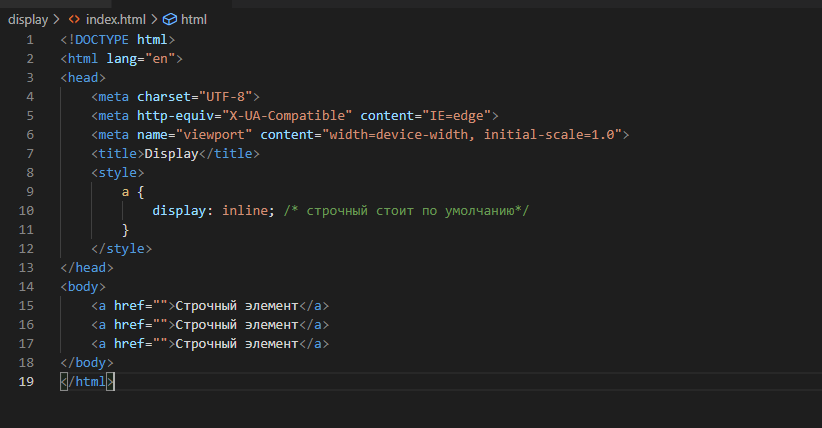
для переопределения типа элемента с блочного на строчный или со строчного на блочный используется CSS свойство display

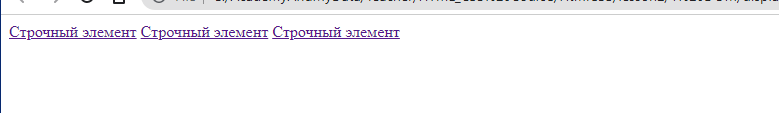
<https://developer.mozilla.org/ru/docs/Web/CSS/display>

<http://htmlbook.ru/css/display>

Пример:

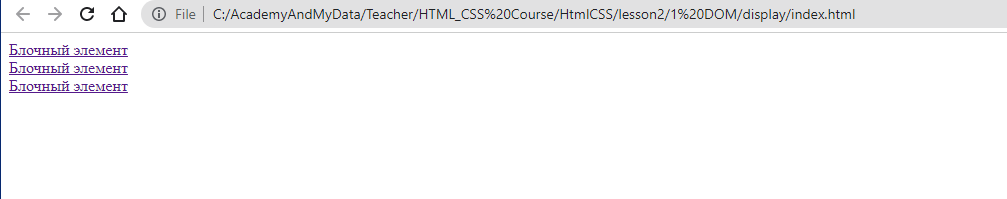
строчный





Магия тег <а> становится блочным





Поддержка браузерами свойство display:

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Значение** | **Описание** | IE6 | IE7 | IE8 | Cr2 | Cr8 | Op9.2 | Op10 | Sa3.1 | Sa5 | Fx3 | Fx4 |
| block | Элемент показывается как блочный. Применение этого значения для встроенных элементов, например тега **<span>**, заставляет его вести подобно блокам — происходит перенос строк в начале и в конце содержимого. |  |  |  |  |  |  |  |  |  |  |  |
| inline | Элемент отображается как встроенный. Использование блочных тегов, таких как **<div>** и **<p>**, автоматически создает перенос и показывает содержимое этих тегов с новой строки. Значение inline отменяет эту особенность, поэтому содержимое блочных элементов начинается с того места, где окончился предыдущий элемент. |  |  |  |  |  |  |  |  |  |  |  |
| inline-block | Это значение генерирует блочный элемент, который обтекается другими элементами веб-страницы подобно встроенному элементу. Фактически такой элемент по своему действию похож на встраиваемые элементы (вроде тега **<img>**). При этом его внутренняя часть форматируется как блочный элемент, а сам элемент — как встроенный. |  |  |  |  |  |  |  |  |  |  |  |
| inline-table | Определяет, что элемент является таблицей как при использовании тега **<table>**, но при этом таблица является встроенным элементом и происходит ее обтекание другими элементами, например, текстом. |  |  |  |  |  |  |  |  |  |  |  |
| list-item | Элемент выводится как блочный и добавляется маркер списка. |  |  |  |  |  |  |  |  |  |  |  |
| none | Временно удаляет элемент из документа. Занимаемое им место не резервируется и веб-страница формируется так, словно элемента и не было. Изменить значение и сделать вновь видимым элемент можно с помощью скриптов, обращаясь к свойствам через объектную модель. В этом случае происходит переформатирование данных на странице с учетом вновь добавленного элемента. |  |  |  |  |  |  |  |  |  |  |  |
| run-in | Устанавливает элемент как блочный или встроенный в зависимости от контекста. |  |  |  |  |  |  |  |  |  |  |  |
| table | Определяет, что элемент является блочной таблицей подобно использованию тега **<table>**. |  |  |  |  |  |  |  |  |  |  |  |
| table-caption | Задает заголовок таблицы подобно применению тега **<caption>**. |  |  |  |  |  |  |  |  |  |  |  |
| table-cell | Указывает, что элемент представляет собой ячейку таблицы (тег **<td>** или **<th>**). |  |  |  |  |  |  |  |  |  |  |  |
| table-column | Назначает элемент колонкой таблицы, словно был добавлен тег **<col>**. |  |  |  |  |  |  |  |  |  |  |  |
| table-column-group | Определяет, что элемент является группой одной или более колонок таблицы, как при использовании тега **<colgroup>**. |  |  |  |  |  |  |  |  |  |  |  |
| table-footer-group | Используется для хранения одной или нескольких строк ячеек, которые отображаются в самом низу таблицы. По своему действию сходно с работой тега **<tfoot>**. |  |  |  |  |  |  |  |  |  |  |  |
| table-header-group | Элемент предназначен для хранения одной или нескольких строк ячеек, которые представлены вверху таблицы. По своему действию сходно с работой тега **<thead>**. |  |  |  |  |  |  |  |  |  |  |  |
| table-row | Элемент отображается как строка таблицы (тег **<tr>**). |  |  |  |  |  |  |  |  |  |  |  |
| table-row-group | Создает структурный блок, состоящий из нескольких строк таблицы аналогично действию тега **<tbody>**. |  |  |  |  |  |  |  |  |  |  |  |

Материалы к уроку:

1. <https://developer.mozilla.org/ru/docs/Learn/CSS/First_steps>
2. <https://developer.mozilla.org/ru/docs/Web/CSS/CSS_Selectors?retiredLocale=uk>
3. <https://developer.mozilla.org/ru/docs/Learn/CSS/Building_blocks/Values_and_units>
4. <https://learn.javascript.ru/css-units>
5. <http://htmlbook.ru/samlayout/blochnaya-verstka/blochnye-elementy>
6. <https://html-plus.in.ua/blochnye-tegi-v-html/>
7. Урок Шаг 2
8. Книга глава 1 Новая большая книга CSS <https://drive.google.com/file/d/11JdRKgw_GE6Tpc3a2RN6ctvqvAuQM7SY/view?usp=sharing>

Домашнее задание:

1. Реализовать сайт визитку. Создать 3 страницы  
   Картинки можно не использовать, но в header и других блочных тегах реализовать background  
   Текст можно использовать Lorem, но если будете использовать реальный текст и придумаете идею какого-то бизнеса будет плюсом. Картинки тоже в плюсе.  
   1 main index.html  
   2 contact contact.html  
   3 about about.html  
   Технологии CSS, HTML  
   Примерный пример:  
   

Автор: Домбровский И.В. “Academy Step”   
Igor Dombrovsky telegram 0990356078

**dombrovskij\_i@itstep.academy**