# Введение

Сегодня мы познакомимся с псевдо-классами, псевдо-элементами и webkit-свойствами. Советую использовать браузер google chrome или яндекс. Поскольку сегодня мы будем писать не кроссбраузерный сайт, то некоторые стили, которые мы будем применять могут не работать в других.

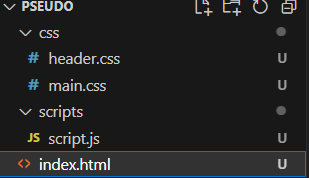
Для более хорошего понимания вам следует знать html, css и немного js. Если с чем-то из этого есть трудности – не беда. Я всё буду подробно объяснять.

Слева сверху у скриншота всегда есть путь и название файла, в котором произошли изменения. Поэтому, если вы запутались и не знаете в каком файле что писать, то обратите на это внимание.

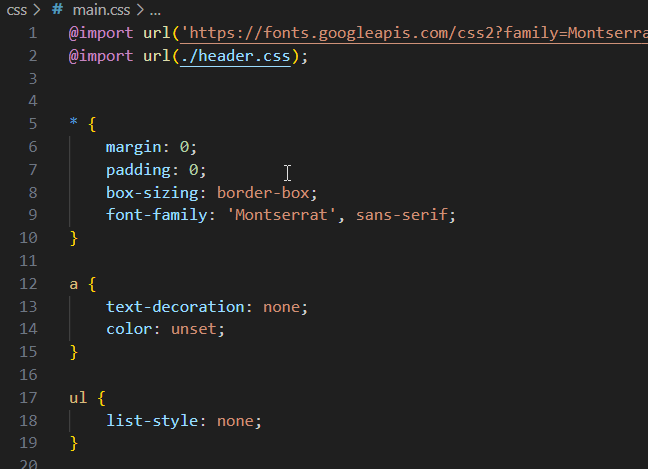
Также по возможности буду оставлять скриншоты с совместимостью на разных браузерах. И помечать их. И ещё прошу обратить внимание на [ссылки](пример%20ссылки), там много можно найти много полезного.

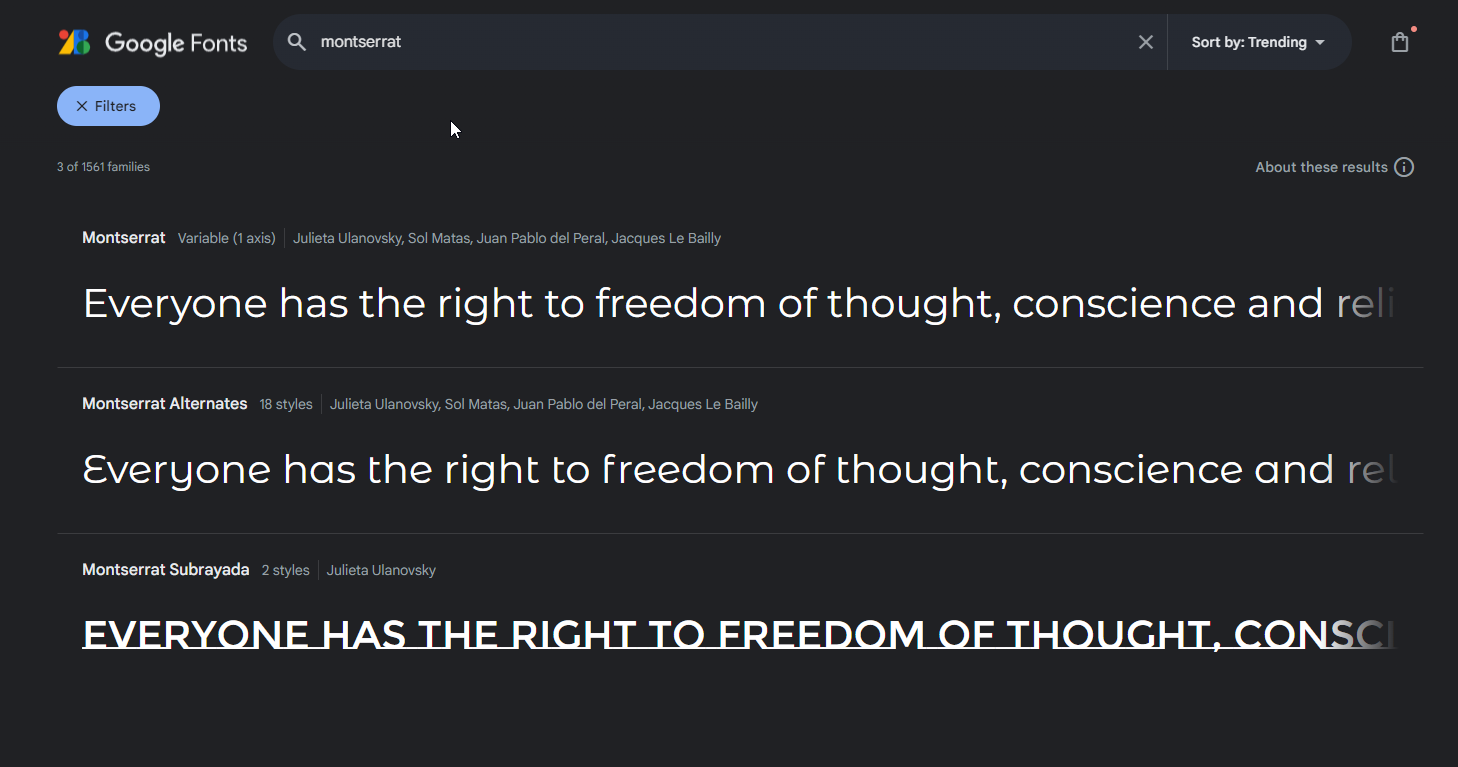
# Начало начал.

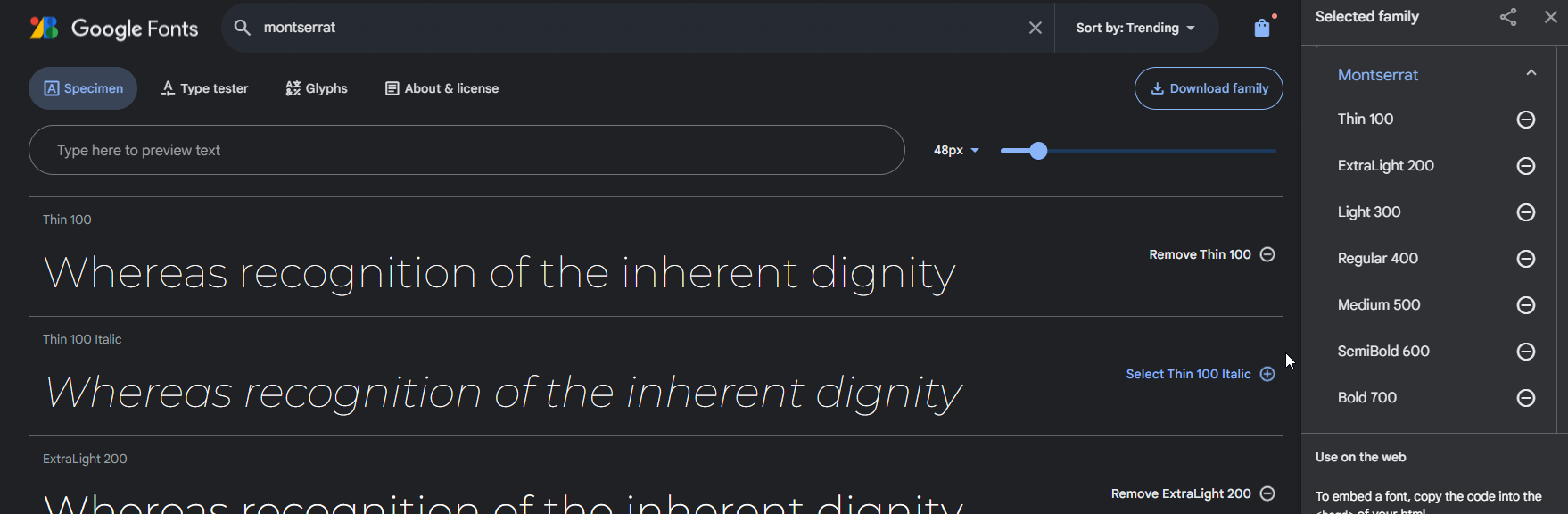
Как и для любого проекта нам нужна пустая папка, название значения не имеет. У меня получилась вот такая начальная структура.



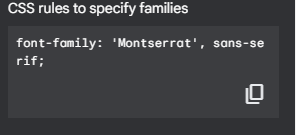
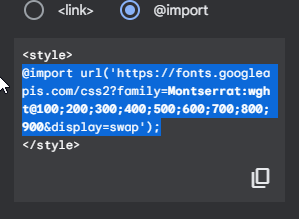
Перед началом любой вёрстки я всегда обнуляю стили. В интернете можно найти css файл с обнулением стилей для всех элементов, но лично мне этого более чем достаточно, будем сюда возвращаться, если понадобиться убрать стандартные стили для чего-то ещё. Также я сразу подключил шрифт и использовал шрифт, советую для этого использовать [Google Fonts](https://fonts.google.com/). Я не стал долго выбирать, решил взять относительно стандартный – Montserrat.

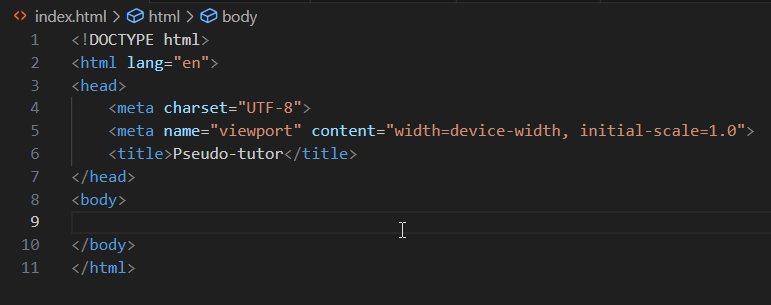




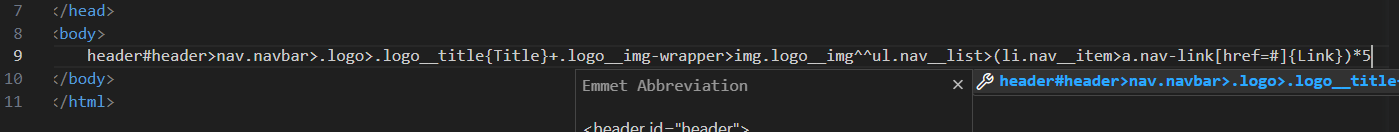


Здесь можно скопировать линк и вставить его в html, но я предпочитаю импортировать шрифт прямо в css-файл.



Теперь можно приступать к вёрстке. Создаём пустой шаблон html. Единственное, что я поменял – title. 

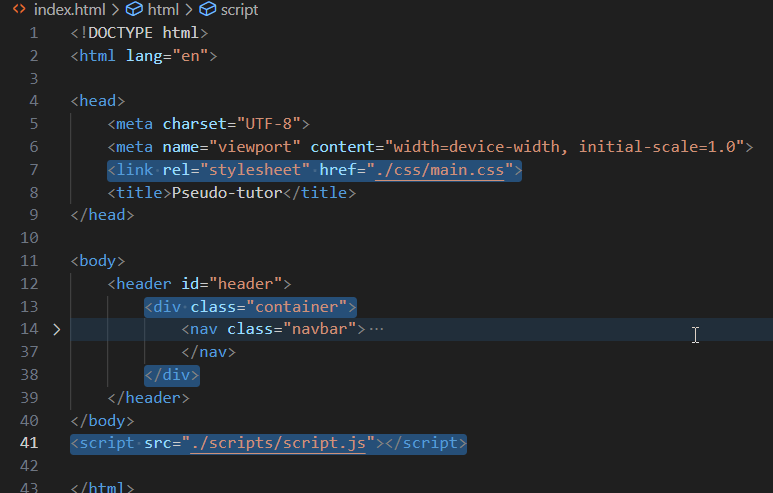
Не обязательно, но на будущее, если не знаете, что такое [Emmet](https://tproger.ru/articles/kak-plagin-emmet-pomogaet-uskorit-rabotu-s-programmnym-kodom/), то советую научиться пользоваться этим плагином. Если упростить - ускоряет вёрстку. Позволяет превратить такую запись:



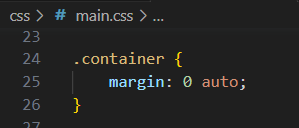
В это.

Быстро пробегусь по моей записи.  
header – указываю какой нужен тег  
# - id этого тега  
> - указываю, что все элементы далее будут вложены в родительский (в нашем случае header)  
nav.navbar – опять же сам тег, но уже не с id, а с классом (если указать просто точку или решётку, то будет создан div с этим классов/id)  
{} – в фигурных скобках текст внутри тега  
+ - показываем, что дальше пойдут элементы на том же уровне вложенности, что и предыдущий  
^ - переходим на один уровень вложенности выше (^^ соответственно на 2 уровня вложенности)  
() – группировка элементов  
[] – указываются атрибуты элемента [название атрибута=его значение]  
\*n - кол-во элементов (n - нужное кол-во элементов)

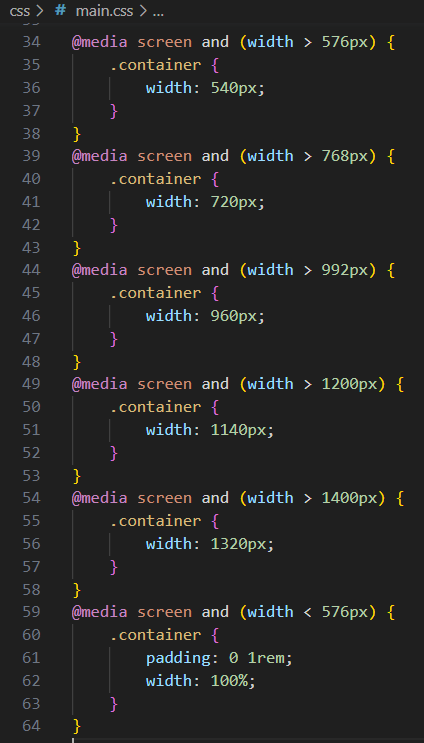
Подключил стили и скрипт (7 и 41 строки). Также обернул весь навбар в контейнер, он нам понадобится, когда будем писать адаптив.



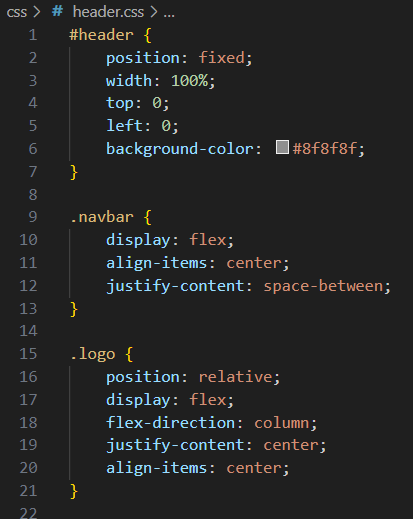
Сразу напишем для контейнера стили. Это для того, чтобы наш блок всегда располагался по горизонтально по центру.



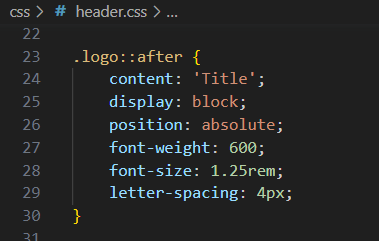
Это для будущего адаптива. Надеюсь, вы понимаете, что тут написано и зачем. Мы говорим, что при ширине экрана > n (в последнем случае < n) к таким элементам применяются такие стили.



Здесь ничего сложного, просто расположил элементы как мне надо. Для хедера задал position: fixed, чтобы при прокрутке страницы он оставался на своём месте. Также временно задал серый цвет, чтобы было видно его границы.



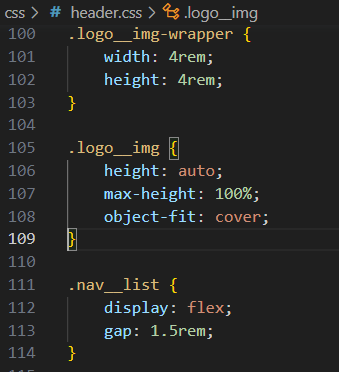
Решил отказаться от подписи рядом с логотипом, лучше сделать анимацию. Тут нам может пригодиться псевдоэлемент [::after](https://developer.mozilla.org/en-US/docs/Web/CSS/::after) или [::before](https://developer.mozilla.org/en-US/docs/Web/CSS/::before). Для них всегда нужно указывать content, то что будет внутри него. Пример: <div>тут контент</div>. Также желательно указывать display, иначе элемент может не отображаться.



Совместимость ::after с разными браузерами

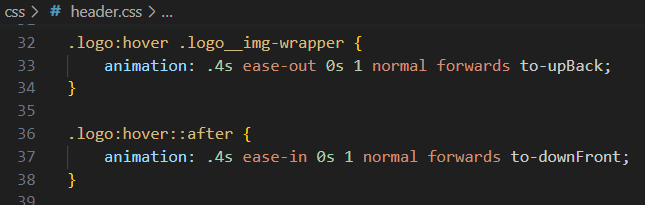


Ужмём наш логотип. И вот уже что-то похожее на привычным header есть.

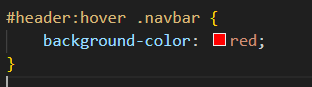


Можно приступать к анимации, которую я задумал, и ради которой отказался от простой надписи возле логотипа. Что-то новенькое. Теперь мы используем псевдокласс. А после псевдокласс и элемент одновременно. Ещё и, возможно непонятное, свойство [animation](https://developer.mozilla.org/en-US/docs/Web/CSS/animation).

Псевдоклассом :hover мы указываем, что следующие стили применятся при наведении на элемент. Запись .logo:hover::after указывает, что при наведении на элемент с классов logo стили применятся к ::after.

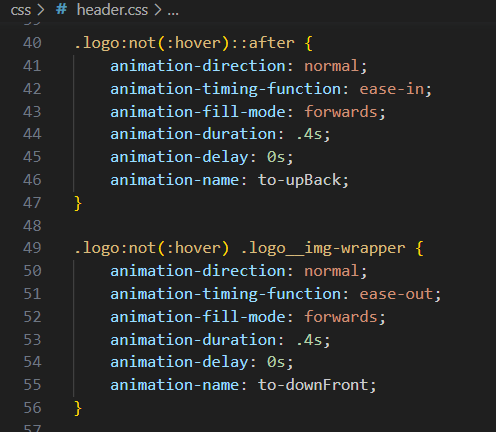


С таким же успехом мы можем поменять цвет у нашего навбара при наведении на header (Делать этого не стоит, здесь просто пример). То есть мы просто применяем наши стили на конкретный элемент внутри родителя, но с условием, что на него (родителя) наведён курсор. After – это псевдоэлемент, .logo для него родитель, поэтому мы и можем так написать и всё будет работать.

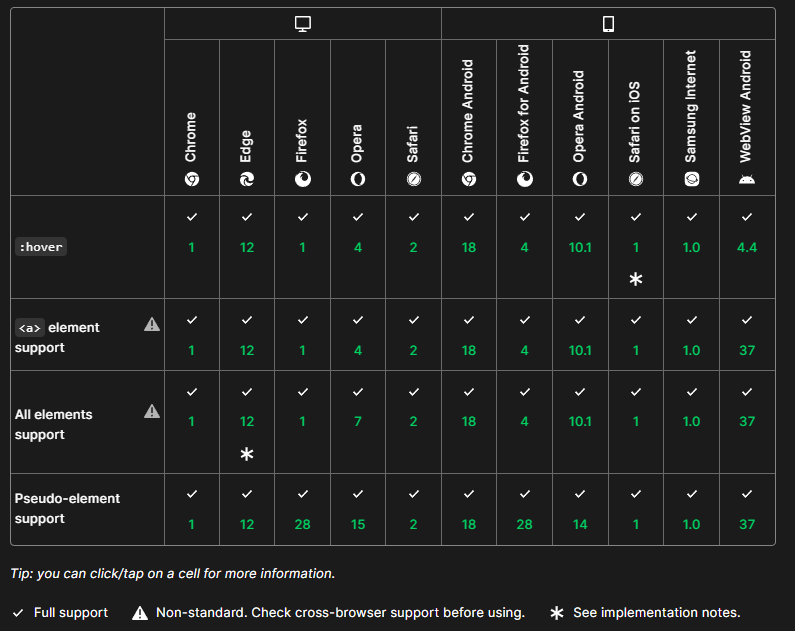


Теперь давайте разбираться со свойством animation. Выше указана короткая запись. Также я написал длинную версию, с указанием каждого свойства отдельно. Но и тут новый псевдокласс - :not(). То есть стили применятся, когда что-то НЕ, в нашем случае не наводимся на элемент с классом logo. Теперь по каждому свойству.

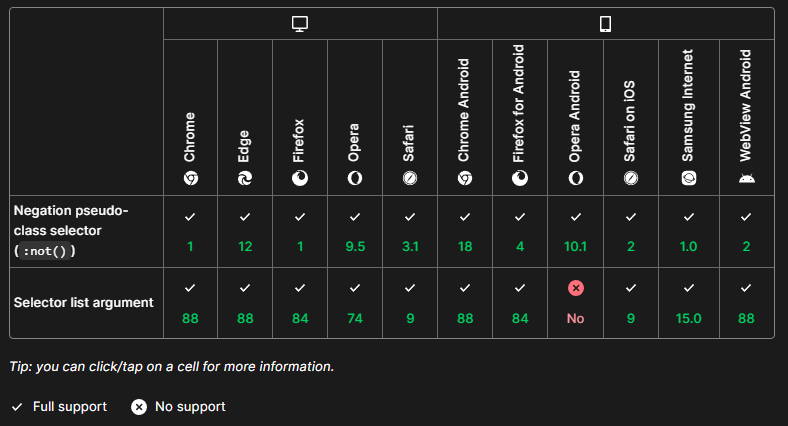
animation-direction – как воспроизводится анимация (вперед, назад или чередуется)  
animation-timing-function – развитие анимации в течении одного цикла  
animation-fill-mode – анимация применяет стили до или после своего выполнения  
animation-duration – продолжительность анимации  
animation-delay – задержка перед анимацией  
animation-name – название keyframe`а, который будет применён



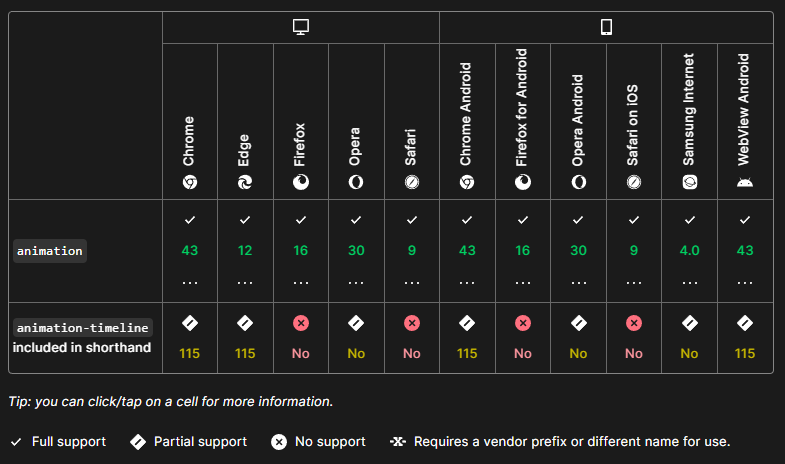
Совместимость :hover с разными браузерами.



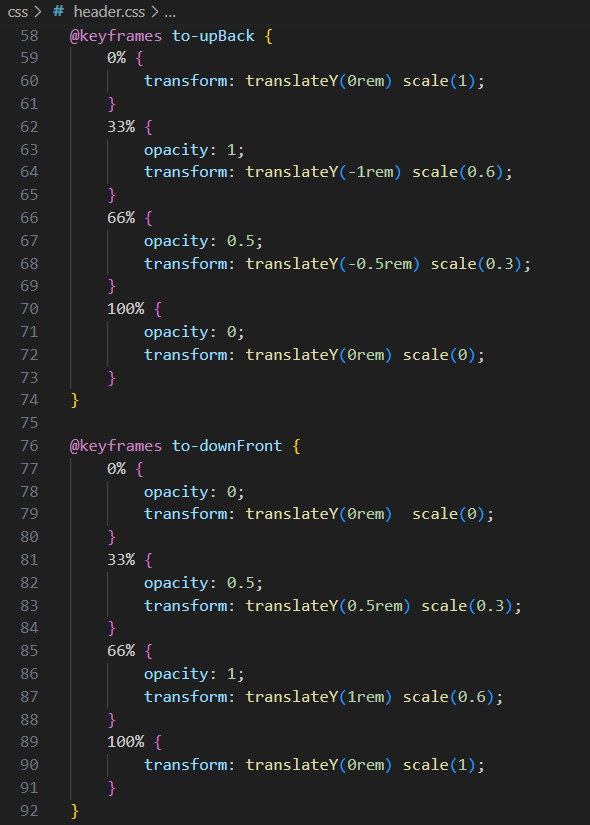
Совместимость :not() с разными браузерами



Совместимость animation с разными браузерами.



Теперь к кейфреймам. ***Ключевой кадр*** (keyframe) - кадр, который содержит конкретные значения тех или иных анимированных эффектов. А теперь своими словами: функция, описывающая поведения элемента в определённый этап анимации. Записываются они вот так:



to-upBack/ to-downFront названия, которые мы указывали у свойства animation-name. Проценты – этап анимации (разрешается указывать любые проценты, хоть каждый отдельный прописать). Можно просто указать from/to, что соответствует 0%/100%. Можете сами поэкспериментировать со значениями, чтобы сделать свою анимацию.  
Сейчас с этой анимацией есть небольшая проблема. Из-за записи .logo:not(::after) она проигрывается при каждом обновлении страницы, что не есть хорошо. Исправим это позже.

Совместимость @keyframes с разными браузерами.



Заменил на более короткую запись. Запись :not(:hover) не имеет смысла поскольку, любой элемент, на котором нет курсора итак выполняет это условие. Но показать, что так можно делать стоило.

