# Вступление  
 you you you you Всем привет, друзья! Добро пожаловать на очередной курс. И давайте я более детально, подробно все подскажу. Естественно, расставим все по полочкам. Давайте для начала разберемся, а зачем нам такой курс. Вроде бы у нас уже есть какие-то знания по HTML и \*CSS\*, и мне кажется, их достаточно. На самом деле, действительно, и HTML и CSS – это достаточно простые технологии. Они не являются программированием, поэтому если вы вдруг хотите, вы можете найти все необходимые моменты, ну и, естественно, создать, будем считать, сайт любой сложности. Тут уже дело, как говорится, в каких-то нюансах практики. Но если говорить про практику, то бывают настолько заковыристые \*решения\*, что нам, понятное дело, нужно в этом всем разбираться. Ну и, честно говоря, если мы разбираем, будем считать, основные моменты, если мы разбираем, будем считать, функционал, который есть в HTML и CSS, он действительно очень объемный. То есть, если вы сейчас зададитесь задачей просто в поисковике ввести, там, не \*знаю\*, HTML теги, вы просто, там, не знаю, крутите барабан, начинаете просматривать это все, очень много всего. Понятное дело, что сначала разбираются основные моменты, потом разбирается то, что нужно, соответственно, в практике, и потом какие-то уникальные составляющие. Вот нашей с вами задачей сейчас является больше практической части. Нам нужно прокачаться именно в таких мощных, интересных моментах по созданию. Именно этой части будет посвящен наш курс. Ну и понятное дело, давайте я для начала представлюсь, подскажу, кто же я такой, почему именно я буду рассказывать про веб-разработку. Зовут меня Кадочников Алексей, возможно, вы со мной уже знакомы. Ну и я буду, соответственно, рассказывать именно про создание сайтов. Созданием сайтов я занимаюсь очень давно, это уже более 9 лет. Ну и самый главный мой козырь, то что я являюсь преподавателем. Я преподаю на портале GeekBrands уже очень давно, ну и получается, что есть необходимый опыт теоретический, есть необходимый опыт практический. Я постараюсь выложиться на 200%, ну и наверняка ждете какую-то прикольную историю. Честно говоря, я замечаю, что HTML, CSS бывает забывается. То есть ты сначала верстаешь сайты, ты переходишь к каким-то отдельным частям, там первое, второе, третье, и потом уже, естественно, начинаешь кодить. Там на JavaScript, у тебя много React, в React есть свои особенности, и ты потихонечку забываешь HTML, CSS. Ну, естественно, когда ты начинаешь верстать какой-то проект или появляется проект на подработке, то тебе нужно знать особенности именно HTML, CSS. И \*знаете\*, прикольная особенность, это то, что ты максимально быстро его вспоминаешь, и я чаще всего по нему скучаю. Поэтому у вас наверняка есть сейчас ощущение, что вы скучали по HTML, поэтому если скучали, устраивайтесь поудобнее и давайте вдохновляться нашим с вами курсом.  
  
# Знать |  
Итак, давайте немножко вдохновимся. Я сейчас хочу продемонстрировать именно первые две части. У нас, естественно, курс не состоит из двух частей. Но я замечаю, что ученики очень часто, не знаю, замечают что-то где-то там в пятом уроке и ждут его больше всего. Ну и, соответственно, упускают первые четыре, а оказывается, что пятый связан с первыми четырьмя максимально сильно.  
  
# Препроцессор |  
Поэтому мы возьмем, будем считать, максимально близкие части, чтобы мы могли работать и с одной, и с другой. Ну и я каждый раз буду добавлять, обновлять эту часть, поэтому я ее назвал не план курса, а именно немного вдохновения. Ну как же узнать, что же нас ждет на протяжении всего нашего с вами курса? Конечно, я подскажу, но не буду разбивать на уроке. Первое, что мы с вами видим, это \*препроцессоры\*. Честно говоря, про препроцессоры чаще всего рассказывается где-то в середине курса. Ну вот опять же там урок 5, например. Если большой какой-то курс или может быть два курса, то чаще всего во втором курсе и то где-то в серединке. И большинство учеников говорят, Алексей, а почему не с первого урока?  
  
# Новшество | Гамбургер-меню | Cs | Знать |  
То есть мне так нравятся припроцессоры, что я хочу с ними работать буквально с первого урока. Поздравляю вас, у нас действительно курс начинается именно с припроцессоров, потому что у нас уже есть все знания и навыки, и нам не хватает именно припроцессорного подхода. Хорошо, следующая часть, и это, конечно же, чего нам с вами не хватает именно в технологиях. Того, что они развиваются. То есть у нас появляется что-то новенькое, что-то вкусненькое. Естественно, я буду рассказывать про \*новшества\* HTML5. Естественно, я буду рассказывать про новшества CSS3. Ну и, например, какие. Естественно, у нас есть элемент Details. Вот вы, возможно, о нем уже знаете. Но на самом деле вы знаете, что он существует. Но вы не знаете, как его стилизовать, о нем уже знаете. Но на самом деле вы знаете, что он существует, но вы не знаете, как его стилизовать, как его добавить. Одновременно с этим вам наверняка интересно, как создается \*гамбургер-меню\*. Можно ли его сделать на HTML, CSS? Вам наверняка очень интересна работа с анимацией. То есть снова же мы можем выйти именно на уровень создания игр. Получается, что этого снова не хватает.  
  
# Абзац  
Итак, новшество HTML5, новшество CSS3. Что еще? Конечно же работа с адаптивом. Адаптивные возможности для нас очень важны. Получается, что этого снова не хватает.  
  
# Ситуация | Cs | Знать | Адаптить | Адаптивность |  
Итак, новшество HTML5, новшество CSS3. Что еще? Конечно же, работа с адаптивом. Адаптивные возможности для нас очень важны. Получается, что мы с вами, естественно, должны добить знания, которые у нас есть, узнать что-то новенькое, вкусненькое. Ну и практически в каждом уроке у нас будет практика, которая наполняется теми же самыми новшествами, теми же самыми особенностями, которые я буду подсказывать. Ну и как я и говорил, на протяжении всего нашего с вами курса будет практика. То есть в теории очень сложно рассказывать. То есть я говорю, что классная особенность, классная HTML, классный CSS, мне очень нравится. Но на самом деле не хватает практических моментов. И я всегда, когда что-то демонстрирую, замечаю, что мне не хватает вот чего-то. То есть рассказываю про HTML, мне не хватает CSS. Рассказываю про CSS, мне не хватает какой-то \*адаптив\* CSS. Рассказываю там вот про CSS, мне не хватает какой-то там адаптивности. Рассказываю про \*адаптивность\*, мне не хватает какой-то плавности. И вот цепляется одно за другое. Сейчас я знаю, что вы уже знаете HTML, вы уже знаете CSS, вы уже знаете про анимацию, вы уже знаете про \*замедление переходов\*. У вас уже есть все необходимые базовые знания и навыки. Чего нам не хватает? Это, естественно, новшеств. Естественно, нам не хватает практики и работы с проектом. То есть в данном курсе я буду создавать проект, я буду добавлять новые возможности. Ну, естественно, мы увидим, как создается. Мы будем считать более что-то такое сложное, интересное и подходящее. Причем проект достаточно знакомый, но это его продолжение. Поэтому, как я и говорил, максимально интересно. Погнали. Знаете, замечаю, что большинство учеников, которые приходят на онлайн-трансляции, они чаще всего говорят мне или пишут в чате, что, мол, Алексей, все, хватит болтать, давайте побыстрее. Всегда хочется быстрее, всегда хочется что-то уловить. Это чаще всего заметно на каких-то бесплатных интенсивах. То есть, всего один урок, и они такие быстрее и быстрее. Они в основном нацелены на трудоустройство. И это нормально. Но, к сожалению или к счастью, мы не можем трудоустроиться там за одно занятие, а то, представьте себе, у нас миллион конкурентов. Через час новый миллион конкурентов у нас в веб-разработке. Поэтому в данной \*ситуации\* у нас действительно последовательный материал. Да, я в первом занятии буду чуть-чуть больше болтать, но чем больше мы, как говорится, узнаем теоретических моментов, тем больше мы можем проявить себя в практике.  
  
# Итог | Cs | Рекомендовать | Sa | Расширение | Знать | Препроцессор |  
Итак, что же у нас сегодня на уроке? Мы сегодня будем разбираться, что такое препроцессоры. Возможно, вы о них уже слышали. Возможно, вы слышали что-то такое, там, применяются, не применяются. Вот я сегодня смогу показать именно на практике, почему стоит использовать препроцессоры. Естественно, мы проговорим весь цикл. То есть, я подскажу, как устанавливается \*препроцессор\*. Честно говоря, я эту тему не думал акцентировать. Думаю, ладно, и так можно написать какой-то лонгрид, вы прочитали, что-то установили. Но замечаю, что так много ошибок, поэтому я решил простой способ. Полностью удалил \*Visual Studio Code\*, все \*расширения\*, все чисто, как говорится, компьютер. Просто там были расширения из предыдущих курсов. И получается, что я сейчас смогу в режиме реального времени продемонстрировать, что я делаю, как я делаю, и действительно, как у меня это все работает. Что нам нужно разобрать? Конечно же, это припроцессор \*SAS\*. Почему? Потому что он очень популярный, ну, наверняка вы именно про него и слышали. Но одновременно с этим есть конкуренты. Честно говоря, они, ну, так себе. То есть, бывает кто-то лучше, бывает что-то хуже, но здесь есть простое детское сравнение. Вот представьте себе, вы сели за руль автомобиля, неваж ну, так себе. То есть, бывает кто-то лучше, бывает что-то хуже, но здесь есть простое детское сравнение. Вот представьте себе, вы сели за руль автомобиля, неважно какой марки. Естественно, вы научились водить этот автомобиль, возможно, вы там, не знаю, ездили на нем два года и теперь решили обновить автомобиль. Вышло что-то, например, новенькое, вы, соответственно, приобретаете новый автомобиль, нужно ли вам заново учиться? Ну, я подразумеваю, скорее всего, не нужно. Бывают ли отличия? Конечно, бывает. Например, вы ездили на каком-то одном автомобиле, вы пересаживаетесь в другой, и там переключение передач не рычаг где-то здесь, а вот под рулем. Удобно ли это? Мне кажется, удобно. Но привычно? Нет, непривычно. Но перейти с одного на другое будет достаточно быстро. Вот здесь то же самое. То есть, я подскажу, что есть один припроцессор, есть другой, и перейти с одного на другой будет вообще элементарно. Поэтому сможете выбрать для себя любименький, но конечно же \*рекомендую\* SAS. Ну и подведем \*итоги\*. Я продемонстрирую, что у нас получилось, продемонстрирую насколько круто мы прокачали сайт, поэтому быстрее погнали в практику. Ну что, начинаем самую главную важную тему нашего сегодняшнего урока и это конечно же что такое припроцессоры. То есть я замечаю, что чем больше ты изучаешь новых технологий, тем тебе сложнее. Ты изучил что-то, ты потратил энное количество времени, потом такой, а я эту штуку не применяю, а мне эта штука неинтересна, а еще хуже, если она устревает. Прям, знаете, что-то появилось, всем нравится, все такие, ура, это, там, не знаю, сотрет все границы, будет использоваться всеми программистами, а потом оказывается, вообще ее никто не использует, и она никому не нужна. Ты потратил время, ты потратил свои, как говорится, ресурсы, но тебе грустно. Внимание, если мы говорим про препроцессоры, я замечаю, что в 99% случаев ученики начинают их использовать. Ну и по себе смотрю, что очень хочется их использовать. Почему же это такой важный подход, и, может может быть он где-то там замалчивается? На самом деле мы ничего не теряем. То есть давайте сразу договоримся или отметаем принцип, который там есть в университете. Типа забудьте, что вас там учили в школе, мы с вами будем изучать что-то новенькое. Нет, возьмите все, что вы знаете по CSS и внимание, мы это улучшим. Мы улучшим со всех сторон. То есть мы будем смотреть, вот чего здесь не хватает, здесь не хватает, здесь. Вот мы каждую часть можем улучшить, посмотреть. Плюс ко всему, я, знаете, буду вести себя как такой, как ученик. То есть, мне кажется, что ученикам не хватает подсказок вида, вот я зашел в Google, вот я нашел первую ссылку, вот я увидел пример, я его применил. То есть, давайте тогда по такому принципу попробуем изучение. Потому что всегда есть мысли, как будто я там все помню и вы должны. Естественно, если я 10 лет занимаюсь веб-разработкой, естественно, я все помню, это нормально, я каждый день этим занимаюсь. Но вы-то не занимаетесь этим каждый день, и это нормально, что вы не помните. Поэтому давайте гуглить. Я тоже гуглю, я постоянно гуглю. У меня есть огромное количество программистов, которые постоянно сидят в гугле, постоянно сидят с какими-то задачками, поэтому будем применять такой принцип именно в обучении.  
  
# Переменный | Cs | Рекомендовать | Вложенность | Цикл | Функция | Знать | Препроцессор |  
Итак, первое, с чего мы начинаем, это что такое припроцессоры. Определение. CSS припроцессоры – это программный подход к CSS. То есть самое простое определение и одновременно с этим самое понятное. Чего нам не хватает в CSS? На самом деле не хватает того самого программирования. Я уже раз 20, наверное, повторил, что ни HTML, ни CSS не являются программированием. И получается, что если мы с вами будем использовать припроцессоры, то мы совершаем первый шажочек в сторону программирования. Если вы уже знаете программирование, то вам будет проще. Ну и давайте рассуждать. Что у нас есть в программировании, из чего чаще всего начинается изучение программирования? Здесь чаще всего используются \*переменные\*. Ага, при процессорах есть переменные. Если у нас есть переменные, значит мы что-то запоминаем. Если мы что-то запоминаем, то для нас это, опять же, очень удобно. Мы можем эту штуку переиспользовать. Вот он, программный подход к CSS, которого не хватало. Но здесь чаще всего ученики говорят про такую составляющую, что в CSS уже есть переменные. Но, честно говоря, они, во-первых, не суперинтуитивно понятны, а, во-вторых, припроцессоры ничего у нас не забирают. То есть, если вы хотите использовать только переменные, то понятное дело, что припроцессоры подключать нет смысла. Если мы хотим использовать большую часть функционала припроцессоров, кстати, важное уточнение, что большую часть, я не рекомендую на начальном этапе вообще погрузиться и на максималку только все-все-все в припроцессорах, это бессмысленно, то получается, что там уже будут переменные и еще много разных плюшек, с которыми нужно разобраться. В итоге, здесь у нас могут быть и \*вложенность\*, и переменные, и \*циклы\*, \*функции\*, \*математические операции\*, все это возможно с помощью припроцессора. Здесь я бы отдельно отметил такие вещи, как циклы, то есть чаще всего они не рассматриваются, но мы с вами будем создавать такую прикольную штуку не на сегодняшнем занятии, а именно при работе с анимацией. Честно говоря, в анимации есть чаще всего проблемы, то есть у нас есть много элементиков, которые как-то двигаются, и они должны двигаться, например, рандомно. Получается, что мне нужно рандом позиции, мне нужно указывать эти позиции для первого, второго, двадцатого блока. И, соответственно, запустить это все в цикле. Естественно, мне поможет препроцессор. Поэтому первое занятие по изучению препроцессоров. А потом на протяжении всего курса мы будем их с вами использовать. Первое, с чего мы начинаем, это установка препроцессора. Как я и говорил, у меня есть Visual Studio Code, в котором не установлено ничего для припроцессора. Сейчас будет такой лайфстайл. Погнали. Переключаюсь. Что мне нужно? У меня есть открытый браузер, то есть у меня здесь есть подсказки. Опять же, я буду пользоваться поисковиком.  
  
# Ситуация | Cs | Рекомендовать | Sa | Знать |  
Второй вариант, у меня есть Visual Studio Code, и здесь только индекс HTML. Внимание, мы пока не прикручиваем к старому проекту. Я, опять же, подскажу основные моменты, можно ли это сделать и как это сделать. Но в данной ситуации вот мы начинаем абсолютно с нуля. Чистый HTML, ничего здесь нет. Что нам нужно? Нам нужно установить расширение. Опять же я рекомендую заглянуть в лонгриды, текстовые материалы, которые есть к уроку. И там я опять же даже со скриншотами, с какими-то переходами. Но я сейчас буду идти, знаете, так по интуиции. То есть, как ты будешь действовать на работе? Что ты делаешь? Ты переходишь, соответственно, расширение, и тебе нужно установить припроцессор. Внимание! Припроцессор еще не установлен. А что это такое? Это отдельная составляющая. Вот у нас есть с вами HTML-овский файл, у нас есть файл стилей. И отдельно будет файл припроцессорный. Вот наша сейчас задача сделать так, чтобы нашего редактора кода понял, что будет существовать отдельный файл, понял его синтаксис и потом мог уже перегнать в обычный css. Что я делаю? Я ввожу sas. Мы будем с вами использовать припроцессор sas, поэтому я, соответственно, ввожу.  
  
# Сайт | Cs | Sa | Видеть | Знать |  
что мы здесь можем заметить первое это у нас синтаксис то есть нашему редактору коды сейчас непонятен синтаксис нужен опять же автоформат поэтому мы устанавливаем расширение sas супер дальше бегу что он не здесь подсказывает я \*вижу\* life сас компилятор его нам также нужно установить вообще возникает вопрос зачем нам несколько расширений неужели одно не может действовать итак давайте разберемся как устроено все будем считать под капотом у нас есть браузер он для нас в приоритете он должен отображать контент как он действует он берет html-овскую страничку и соответственно вместе с ней приезжает css отлично браузер знает и HTML, и CSS. Вот он на эту штуку заточен. Получается, что мы берем и HTML, и CSS, создаем тот самый прекрасный внешний вид и отображаем его для пользователя. Все просто. Теперь давайте подумаем, если у нас будет третий файл, SAS, например, должен ли его знать браузер? Вот он не должен этого делать. Почему? Да потому что у нас несколько препроцессоров. Завтра появятся новые, чего разработчикам Google Chrome нужно будет подтягивать те же самые знания, улучшать свой как-то там, не знаю, браузер и делать такой функционал. Сколько таких браузеров, сколько такого функционала. Получается, что эта штука у нас бесполезная. И нам нужно сделать так, чтобы из препроцессорного файла у нас получился CSS-файл. Вот и простой принцип. То есть у нас есть препроцессорного файла у нас получился CSS-файл. Вот и простой принцип. То есть у нас есть припроцессорный файл, он перегоняется в CSS, и CSS уже отдается браузеру. Именно по этой причине у нас будет три файла, но мы точно также будем пользоваться двумя. Отдельно HTML контент \*сайта\*, отдельно припроцессорный стиль для нашего с вами сайта. Хорошо, а что именно он будет перегонять, что он будет компилировать?  
  
# Style | Ситуация | Переменный | Cs | Рекомендовать | Видеть | Знать |  
Я только что рассказал, что могут быть переменные.енные мы создали переменные мы добавили в них значение ну естественно эти значения нужно перегнать мы будем ждать функционал нам нужно их добавить по нашему сами сайту где-то цвет фона где-то цвет текста и так далее все добавили переменные получили чистый css именно эту задачу он будет выполнять. Поэтому давайте сразу для себя, если ставите где-то какие-то пометочки, я прямо рекомендую взять, не знаю, лист бумаги А4, ну и просто записывать для себя какие-то подсказки. Самые важные части. Первое. Вам нужно установить расширение все. Второе. Вам нужно понять, что вы работаете с двумя файлами. Это индекс HTML и припроцессорный. В данной ситуации у нас будет \*style\* сас ну или с css я для начала с css буду использовать внимание css файл мы не открываем по крайней необходимости если вы даже открыли css файл вы в нем ничего не пишите и так у нас под запретом пишем под запретом писать что-либо в css отменяется теперь только в при процессоре. Хорошо, нажимаю кнопочку установить, он соответственно его устанавливает, иногда просит о перезагрузке, то есть нормальное явление. Что я дальше вижу? То есть здесь у меня больше никаких отдельных подсказок нет и бывает знаете что в Он мне бывает пишет live. Live уже есть. У нас автозапуск. Авторан, как-то так. Лайфсервер, по-моему, так называется.  
  
# Абзац  
Итак, лайфкодинг, лайфсервер.  
  
# Абзац  
Итак, лайфсервер, что это означает? То есть сейчас будет запускаться, будем читать эмуляции.  
  
# Абзац  
И по факту это очень важный инструмент, когда вы начнете работать с тем же самым JavaScript. Поэтому, возможно, у вас эта штука уже установлена.  
  
# Sa | Livesas | Cs | Расширение |  
Итак, у меня три расширения. SAS для поддержки синтаксиса в редакторе кода. Второе расширение. Так, здесь не требует перезагрузки. Хорошо. Второе расширение – это \*LiveSAS\* компилятор. То есть мы перегоняем из SAS в CSS. Ну и третья составляющая – это \*Live Server\*. То есть чтобы эта штука могла происходить, будем считать, в момент работы с кодом. То есть вы что-то пишете в препроцессоре, и, соответственно, это автоматически перегоняется в CSS. Хорошо, что я могу заметить?  
  
# Style | Cs | Sa | Писать | Sc |  
Я здесь могу увидеть Go Live, ну и, соответственно, как это будет работать. Если я нажимаю live, он мне запускает эту HTML-овскую страничку, и теперь ее даже сохранять не нужно. То есть, ну давайте проверим. Нажал live, он starting, запустился в браузере, и я ничего не сохраняю. Так, какие-то подсказочки, ничего страшного, не подсказывай. Я что-то ввожу, ничего не нажал, не обновляя страничку Вот опять же появляется значение То есть мы сейчас с вами работаем с сервером И это нормально То есть вы можете запускать какие-то крутые штуки из JS Но опять же эта часть нас не очень волнует Хорошо, следующий этап Где припроцессор? Ну давайте как обычно добавим структуру HTML документа И нашей с вами задачей является создать припроцессорный файл. Чем он будет отличаться? Он будет отличаться только расширением. То есть здесь не нужны танцы с бубнами, не будет такого, что я сижу, думаю, ой, что же делать? Возможно, придется установить какое-то еще расширение, но мы пока действуем по инструкции. Ctrl-N, Ctrl-S. Я здесь \*пишу\* style-s css и нажимаю сохранить. То есть припроцессорный файл. Я здесь пишу style.\*scss\* и нажимаю сохранить. То есть припроцессорный файл. Смотрим, LifeSize Compilator, новые настройки. Отлично. Есть значение, он говорит, что, соответственно, все работает. Я создал припроцессорный файл. Давайте посмотрим, что у меня есть. У меня есть два файла. HTML и припроцессор. Внимание, я сказал, что файлов будет три. Где третий файл? Где файл стиль? Ну и первое, что нам необходимо, это проверить, что все работает. То есть, естественно, я хочу, чтобы у меня был HTML. Что-нибудь добавим. Давайте h1, тестируем. SCSS. Внимание, не SAS, а SCSS. Я подскажу, чем отличается.  
  
# Watchsas | Cs | Sa | Видеть | Знать | Sc | Color |  
Я подскажу, как это опять же все можно настроить Но для начала SCSS Внимание, расширение SCSS при процессор SAS Нормально Тестируем SCSS Ну и, как обычно, я добавляю класс Title Дальше, при процессоре мне нужно добавить стиле Точечка Title Ну и давайте там текст line center \*Color\* red Точно, чтобы мы были уверены Внимание, все один в один То есть не в смысле что-то по-другому Не в смысле что-то новое Те же самые селекторы Те же самые фигурные скобочки Те же самые текст line center И так далее Абсолютно все то же самое То есть если у вас есть какой-то супер большой проект Вы хотите начать использовать припроцессоры сегодня Вы просто все содержимое из CSS копируете Вы добавляете его в style.scss И это будет работать Главное, чтобы у вас там не было ошибок Естественно, если нет ошибок, тогда это заработает Итак, все так же два файла Где третий? Где наш CSS? Браузер сейчас не отобразит. Давайте проверим. Так, эту штуку мы не используем. Нас интересует индекс HTML просто из Finder. Тестируем CSS. Внимание, стили не работают. Почему? Потому что файла стилей нет. Он не может отобразить контент, он не может сам догадаться, что есть при процессоре. Что для этого нужно? Вот здесь есть такая кнопочка \*WatchSAS\*. Внимание, ее нужно нажать один раз при работе с проектом. Я сегодня открыл проект, я нажал WatchSAS, я с ним, соответственно, работаю. При нажатии он мне должен скомпилировать, то есть создать самостоятельно файл стилей. Ну и, соответственно, этот файл стилей и послужит для индекса HTML, опять же, будем считать результатом. Из уточнений. Каждый раз эту кнопку нажимать не нужно. Нажали, включили. Нажали еще раз, выключили. Получается, что вы нажали один раз, работаете с проектом. Закрыли проект, там, не знаю, легли спать, проснулись, утром начинаете, открыли проект, нажали. То есть каждый раз включаем обработку припроцессора. Нажимаю watch sas. Что я вижу? Первое, что меня там пугает, это терминал. Не надо его бояться. В нем есть все подсказки. Это тот самый, там, не знаю, отладчик, только в редакторе кода. Что я здесь вижу? Он мне подсказывает, что все сгенерировалось. Все супер. Если у меня здесь какие-то проблемы, то есть он, например, чего-то не знает, какие-то ошибки, то он мне будет писать. Эррор. Какие-то у тебя там фигурные скобочки. Я не понимаю. На первой строке фигурная скобка. Первая строка, давай, исправляй быстрее. То есть прямо здесь все подсказки. О, так это круто. До этого в CSS, не знаю, можно было просто скопировать HTML, туда добавить. Он такой, ну, не проблема. Все в порядке. Сейчас любая строчка компилируется и говорит, что ему не нравится. В итоге, точечка title. Внимание, я ничего не нажимаю.  
  
# Cs | Смотреть | Знать | Sc | Препроцессор |  
То есть, нет такого, что я нажал Ctrl-S и так далее. Он мне скомпилировал и говорит, все работает. Давайте проверять. А, я не подключал, кстати, файл стилей. Он все равно не заработает. У меня появился стайл CSS. Внимание, я его не открыл. Вот \*смотрите\*, у меня нет открытого файла стилей. Я его не трогаю. К нему нельзя прикасаться. Его сгенерировал препроцессор. В индекс HTML я подключаю файл стайл CSS. Браузер не знает о том, что у меня припроцессор scss. Я в конечном итоге вообще могу стереть все, что связано с припроцессором. Ему важен css файл. Css файл сгенерировался, я его подключил. Ну давайте теперь проверять. Обновляю страничку. Во-первых, текст по центру красного цвета. Смотрите, я установил абсолютно с нуля препроцессор. Ничего у меня здесь, не знаю, не лагало, ничего не было установлено, предустановлено. То есть ничего я не перезагружал. Ну и, естественно, у нас появилась компиляция кода, и мы получили нужный нам CSS. Возможно, возник вопрос, что это за четвертый друг-товарищ. Ну, на самом деле, мэп, такая как карта, ну это просто подсказки. друг-товарищ. Ну, на самом деле, мэп, такая как карта, ну, это просто подсказки. То есть, у нас есть там версия припроцессора, какой файл получился, какие отдельные там значения. То есть, да, мы можем это все менять, да, мы можем это все располагать в отдельные папки. Здесь как хотите. Ну, вот, опять же, у нас там три файла.  
  
# Смотреть | Cs |  
В идеале уже создавать папку там какой-нибудь CSS, внутри нее это все размещать и, соответственно, с этим работать. Здесь уже смотрите сами.  
  
# Ситуация | Cs | Рекомендовать | Смотреть | Знать | Sas | Препроцессор |  
Я рекомендую такую штуку сделать, если больше это все размещать и, соответственно, с этим работать. Здесь уже смотрите сами. Я рекомендую такую штуку сделать, если больше файлов. То есть мы создадим, не знаю, файл отдельно для страницы каталога. Рекомендую. Создадим отдельный файл, например, для медиазапроса. Тогда рекомендую. В ситуации, когда всего два, три файла, то не рекомендую. Хорошо, мэп не открываем, CSS не открываем. Мы работаем все так же с двумя файлами что делать вот с этой вот будем считать объемной частью я рекомендую разместить ниже и и также можно закрыть нет такого что компиляция прекратилось все так же работает она выскочит снова когда у вас появится ошибка поэтому можно в принципе так использовать файл при процессорный ничем не отличается от css поэтому по факту у вас такое же разбиение, такой же внешний вид, поэтому отличий вы не должны заметить на начальном этапе вообще никаких. Поздравляю, припроцессор установлен, можно и нужно пользоваться. На данном этапе я рекомендую опять же все это проделать и не упустить ни один из шажочков. Потому что там, не знаю, 99% вопросов Алексею у меня не работает. Смотрю, забыл нажать там watch \*sass\*, забыл расширение файла, забыл там, не знаю, установить какое-то расширение, забыл, что подключается файл стилей. Все по шагам. Внимание, у нас, во-первых, все прописано, во-вторых, все записал для вас видео. Поэтому перепутать или создать как-то по-другому шансов не будет. Ну и, естественно, не будет порога вхождения. На данном этапе отличий у нас ровно ноль. Ну что, мы установили с вами препроцессор, мы его запустили, мы увидели, что все работает, и отличий ноль. Ну и первый, наверняка, возникший вопрос, а что себе вообще в принципе представляет препроцессор?  
  
# Ситуация | Переменный | Cs | Рекомендовать | Смотреть | Видеть | Препроцессор |  
Итак, при знакомстве с препроцессором первое, что хочется отметить, что данный препроцессор является одним из самых популярных. То есть мы ориентируемся на то, что это не единственный преппроцессор, но он опять же топчик по популярности, поэтому мы его используем. Хорошо, в итоге активная у него поддержка, активно развивается, ну и соответственно очень большой функционал. То есть это один из таких более молодых припроцессоров и одновременно с этим он учел ошибки предыдущих, подтянул, как говорится, функционал нужных и, естественно, используется те самые возможности, которые нас интересуют. Что за возможности такие? Смотрите, я собрал список. Честно говоря, опять же, можно выносить отдельное определение и тогда опять же затянется урок, тогда будет куча теории и практически мы не увидим результат. Поэтому мы сейчас с вами разобрали препроцессоринг, то есть мы понимаем, что это такое. Отдельный файл, генерируется CSS, он нужен для того, чтобы писать на программном языке именно в препроцессорном файле, а потом эта штука перегонится в обычный CSS. Но что за программерские особенности? Первое, с чего я начал, это, конечно же, переменные. Вот именно с переменных мы с вами и начнем. Честно говоря, я не очень люблю такие списки. То есть, когда я вижу перед глазами в лекции где-то список, я понимаю, что мне его или заучивать, или переписывать. Но в данной ситуации рекомендую вот как поступить.  
  
# Сайт | Переменный | Рекомендовать | Sa | Смотреть | Видеть | Писать | Знать | Color |  
Итак, начинаем с переменных. Ну вот, если вы забыли. Вы переходите, соответственно, в браузер. И здесь просто в поисковой строке, показываю, вводите SAS. То есть, нас единственное, что интересует. И здесь есть документация. Документация, соответственно, есть на английском, есть документация на русском языке. Выбирайте для себя. На русском чаще всего есть реклама, хотя и на английском тоже существует, но смысл, как говорится, в том, как вам комфортно. Первое, что я здесь могу увидеть, это, соответственно, меню сайта. И в меню сайта я вижу «Изучить SAS», «\*Learn SAS\*». Ну и получается, что здесь сразу написано, что такое припроцессоринг. То есть, с чего мы с вами начинаем, как это все запускается. И опять же, первая составляющая переменная. Поэтому, если вы забыли припроцессор, вам не нужно искать какие-то лекции, не нужно искать какой-то список. Просто зайдите в официальную документацию, здесь есть примеры. Понятное дело, что у них есть свои примеры, у нас будут свои примеры. Я рекомендую, опять же, вот смотреть, где используются те же самые переменные именно в проекте. То есть, да, можно создать переменную и сказать, ребята, смотрите, он ее запомнил, вот какой молодец. Но мы с вами будем, опять же, смотреть более обширно. Что мне нужно? Первое, что я делаю, это я открываю макет. И сейчас мы можем порассуждать, что можно добавить в переменные Какие основные моменты, опять же, меня интересуют Как правильно это записывается и как правильно это используется Ну что, надеюсь, такой подход вам, опять же, нравится Поэтому давайте смотреть Ну что, знакомый наверняка проект И первое, что я вижу Вот изначально я \*выцепляю цвет\*, который у нас чаще всего встречается. Опять же, на каждом сайте есть свои какие-то уникальные особенности. Если сайт не черно-белый, то, скорее всего, у него есть какой-то селективный цвет. Понятное дело, что у нас дизайнер, может быть, любит попугаев, он добавляет там просто какую-то радугу. Такой вариант мы не рассматриваем. Что я вижу именно в этом сайте? Здесь мне заметен вот такой вот розовый цвет. Вот именно этот селективный цвет для нас опять же в приоритете вот он у логотипа вот он у нас соответственно у количество там элементов товаре вот у нас активная страничка на странице каталога название нашего там блока здесь цена выделена этой составляющей Опять же, прокручу, вот они Те самые значения, которые встречаются очень часто Эффекты наведения Получается, что мы с вами заметили селективный цвет И его лучше всего добавить в переменную Ну а зачем? Зачем мне это делать? То есть, а в чем преимущество данной переменной? Ну что, давайте я продемонстрирую Мы будем создавать что-то подобное Опять же, title пока убираем Заголовок пока убираем Видите, у меня появилась часть снизу Он говорит, пустой файл. Что-то пока будем смотреть, как все работает. И я в припроцессорном файле сейчас пишу значок доллара. Это классная составляющая. Мы сейчас будем еще разбирать припроцессор лес, и в нем действительно другой синтаксис, и он неудобный. Доллар очень удобный. Придумываем название. Название переменных я рекомендую писать в camel case, рекомендую писать на английском языке. Начинается все со строчной буковки. Ну и, соответственно, логично. Не экономьте буковки. То есть вот эти вот, знаете, история типа color. А что за color? Что за элементы? Давайте выставим color select site. То есть я подразумеваю, что селективный цвет на сайте то есть тот самый активный ну вот опять же называть как хотите дальше ставим двоеточие и выставляем то самое значение ну естественно я могу нажать на любой из блоков давайте там опять же нажимаем заголовок вот я вижу это значение мы будем его брать из инспектирования и так я скопировал данное значение и добавил сюда ну естественно хочется понять где это применяется. То есть видно ли это будет на страничке. Давайте создадим что-нибудь. Прямо этот блок.  
  
# Ситуация | Сайт | Cs | Рекомендовать | Смотреть | Писать | Color |  
Итак, у нас сначала идет хедер. Я потом добавлю наполнение. Хедер, хедер, семантические элементы. И потом у нас будет в данной ситуации это часть шапка каталога. Новые предложения. Я вообще просто создам блок. Давайте Top Heading. Top Head, как голова. Мне кажется, так даже интуитивно понятно. Верхняя часть, голова, будем считать, у нашего с вами сайта. Отлично. Здесь у нас есть какой-то заголовок. Я копирую данный текст и добавляю его сюда Понятное дело, что внутри элементом, например, h2 Я пишу top head heading То есть есть верхняя часть и у него есть заголовок Хорошо, немножко стилистики для top head heading. Рекомендую опять же копировать такие части, если у вас нет подсказок из CSS, то прям очень подходящая составляющая. Ну что, берем мы с вами стилистику, мы копируем опять же всю-всю-всю эту часть, добавляем ее сюда. И смотрите, что я делаю. Вот у меня есть переменная. И здесь я пишу решеточка color select site. Внимание, мне даже подсказал эти основные моменты. И работает автоформат кода. То есть форматирование CSS у меня сработало. Из уточнений. Вот на одном примере незаметно, даже не так. В целом мы увидим, что цвет должен поменяться. Проверим. Мало ли, может не поменялось ничего. Проверяю. О, да, действительно есть такая составляющая. Но это плохо видно в ситуации, если у нас несколько блоков.  
  
# Ситуация | Смотреть | Link | Видеть | Писать | Знать | Color | Breadcrumb |  
Итак, дальше. У меня есть навигация. Брэд Крампс. Вот смотрите. Частая проблема. Я не знаю, как пишется Брэд Крампс. Я представляю. Брэд Крампс. В принципе, Брэд Крампс. Как-то так.. Я не знаю, как пишется \*breadcrumbs\*. Я представляю breadcrumbs. В принципе, breadcrumbs как-то так. Но я не знаю. Я знаю, что это хлебные крошки. Переводчик. Переводчик. Хлебные крошки. Хлебные крошки. Breadcrumbs. Все. То есть, правильное название классов не надо придумывать, не надо здесь что-то из себя представлять. Breadcrumb. Хорошо. Внутри у нас соответственно есть несколько ссылок. Я пока без разделителей. Давайте у нас просто будет там Home, меню и новые предложения. Это просто будет несколько ссылок через пробелы. Снова же лайфхак у нас курс более профессиональные более правильный подход вот я понимаю что мне нужно сейчас стилизовать сразу три элемента три элемента одинаковая как я поступаю в данной ситуации я зажимаю клавишу alt в некоторых редакторах года это control так что если они работают попробуйте control Visual Studio Code это клавиша alt. Если у вас macOS, то это клавиша option. Дальше я пишу a, это ссылка, и переписываю breadcrumbs. Все, есть такое. \*Breadcrumbs \*link\*\*. И, соответственно, выбираю. Смотрите, у меня сейчас создалось сразу же три ссылки. Они у нас пока никуда не ведут. Кстати, вот home это главное тоже. Она должна вести на главную. И что я делаю? Я зажимаю клавишу shift, выделяю закрывающийся тег, вырезаю его, control x, прыгаю в конец строки. В macOS это command вправо, в Windows это чаще всего кнопочка где-то в правом верхнем углу and и нажимаю control в смотрите я сейчас создал соответственно отрисую у нас есть уникальная это у нас ссылка нашей странице поэтому я бред крамплинг сайт пишу или но опять же там страничка думайте как вам удобнее вот у бред кр \*Breadcrumb Link\* у нас есть стилистика. Так, давайте проверять. Для Breadcrumb Link мы выставляем стили, которые у нас здесь представлены. Проще всего выделить нужный текст и посмотреть стили. Вот он, серенькое значение, все круто. Копирую все, кроме разновидности шрифта. Отлично. Удаляем комментарии, которые не нужны. И у нас есть уникальный элемент. Давайте посмотрим, чем он отличается. Ну, я вижу, что он попухлее выглядит, да? И у него другое значение цвета. Вот font-weight 300, а здесь у нас font-weight 700. Ну, вот они отличия. Я копирую только отличающиеся части. Так, breadcrumb link site. И для этой штуки мы выставляем только отличающиеся части. Все остальное совпало. Внимание, значение цвета мы видим, что один в один, как у нас, доллар, color, site. Давайте проверять. Обновляю страничку, вижу ссылки. Нужно убрать нижнее подчеркивание. Давайте сразу же будем делать красиво.  
  
# Видеть | Переменный | Знать |  
Итак, A, текст Decoration No. Есть нужные блоки. Чуть-чуть приближу для наглядности и значения цвета. И вот сейчас переменные заиграют новыми красками. Представьте себе, что я создал весь проект. Вот эти вот все значения. Все покрасил в такое значение цвет Прибегает дизайнер и говорит Нет, некрасиво Мы решили менять какое-то свое У нас полностью ребрендинг Мы хотим стать более брутальными Давай у нас будет не розовенький, а фиолетовенький цвет Там сказали Отлично, мы выбираем новое значение цвета Я просто не могу сейчас выбрать какой-то отдельный фиолетовый Не знаю назв название, поэтому я сейчас буду выбирать, например, такой ближе к синему. Выбрал значение переменной. Внимание, я ничего не менял где-то там отдельно, там же у нас переменные. И я обновляю страничку. Что я вижу? Везде стало фиолетовое. То есть, представьте себе в масштабах того же самого интернет-магазина. Таких вот вариантов селективного цвета вагон. То есть, если это будет задачка для меня без припроцессоров, я просто расплачусь, я пойму, что мне нужна неделя. Я наверняка это все попытаюсь поменять, но у меня не получится. Ну, почему не получится? Скорее всего, получится, но я, может быть, что-то забуду. Я, может быть, не захочу это делать.  
  
# Сайт | Ситуация | Переменный |  
Я не хочу тратить на это время и так далее. Всего одна переменная. Естественно, в переменные не \*обязательно добавлять\* только значение света. То есть мы можем добавлять и значение, ну, например, даже рамок, то есть 1 пиксель, solid, там, black, например. Мы можем какие-то выражения отдельные. Мы можем, ну, давайте опять же смотреть со значением ширины. Наверное, мы возьмем ширину нашего сами сайта. Внимание, ширина сайта это не вот вся простыня, не 1600 пикселей. 1600 пикселей может быть, но в данной ситуации это ширина контентной части. Вот я выбираю контентную часть и смотрю, это 1140.  
  
# Header | Сайт | Width |  
Итак, я создаю еще одну переменную. Dollar \*width\* site. Ширина нашего сами сайта 1140 пикселей. Ну и давайте для блока \*header\*, например, просто зададим высоту и ширину приплюсуем такую, только для наглядности.  
  
# Сайт | Смотреть | Header | Писать | Знать |  
Итак, я пишу header, высота 100 пикселей и ширина доллар ширина сайта задам рамочку просто чтобы мы его увидели и опять же поймем как это работает итак я создал шапку сайта вот он прямоугольничек вот она ширина внимание мне это значение действительно необходимо потому что может быть сетка меняется это снова зависит от дизайнера это снова бывает ситуациях когда знаю, у нас старый какой-то сайт. Он использовал еще старое разрешение экрана, и там было там, не знаю, 800. Ну представьте себе, это неудобно. Я работал с n-ым количеством проектов, с n-ым количеством тех же самых лендингов, которые становились больше. 1140 это стандартное такое значение, но в среднем где-то там 1200 крутится это все. Ну потом они решили, что им нужно больше. Они хотят, чтобы контент расползался, потому что слишком все как-то компактно. И они поменяли это значение на 1440. Вот смотрите, что будет для меня, если я для всех блоков поменяю значение ширины. Я действительно буду опять же расстроен.  
  
# Абзац  
Если мы берем переменную, у нас одна переменная, с этим значением можно и нужно работать.  
  
# Абзац  
Внимание, мы можем с этим значением работать даже с помощью математики.  
  
# Знать |  
Поэтому ни в коем случае не подумайте, что мы сейчас зафиксировали значение ширины, что у нас там, не знаю, появится полоса прокрутки и так далее. Нет! Мы можем эту штуку использовать для нашего блока центра.  
  
# Знать |  
Мы можем эту штуку использовать для разбиения, там, не знаю, на три равные части и так далее. Все опять же смогу продемонстрировать, когда перейдем к математике. Ну что, самая главная часть, которая нас с вами волнует, это блок-центр. И, честно говоря, здесь все намного хитрее. Я бы даже сказал, совсем непонятно.  
  
# Padding | Сайт |  
Итак, мы знаем, что создается блок-центр для центрирования проекта. Пишется \*padding\*-left, calculate, и здесь пишется 50%, ой, 50 долларов тоже неплохо, но лучше в процентах, минус ширина нашего с вами сайта пополам. Вот мне сейчас нужно взять ширину сайта. А, у меня есть переменная, ничего себе. Давайте ширина сайта. И я ее делю пополам. Ну, внутри у нас калькулятор вроде бы должен сработать правильно. Сначала он должен разделить, потом вычесть, потом получить результат. Я добавляю, копирую. Здесь значение right. И проверяем. Ну, добавим куда-нибудь сюда. Центр.  
  
# Header | Ситуация | Сайт | Видеть |  
Итак, наш с вами блок header. Ну, во-первых, я вижу, что его разнесло. Проверяем, просмотреть код, смотрим, что у нас в отладчике. Меня больше всего интересует блок центр и вот они 570 пикселей. Ничего себе! То есть он сейчас взял переменную, он просто разделил ее пополам и вывел результат. Внимание, бывают такие ситуации, когда эта штука не работает. И тут как в JavaScript есть как экранирование. То есть там пишется такой амперсант, фигурные скобочки и значение сюда. По-моему не амперсант, да. Что мы можем заметить? Что внутри фигурные скобочки и значение сюда по-моему не амперсант да что мы можем заметить что внутри фигурных скобочек сейчас выдает ошибку то есть ему не нравится такой подход поэтому мы используем стандартное значение в математике он взял ширину сайта разделил ее пополам получил \*570 1140\* разделить пополам это 570 и подставил значение сюда поэтому никакого значения ширины нам здесь не нужно ведь ширину мы с вами убираем и давайте от лачек чуть подвинем здесь немножечко отдаление обновили страничку и так смотрим контент остался по центру идеально работает то есть я в данной ситуации не то что не проиграл я выиграл если у меня завтра поменяется значение ширины и станет 1440, представляете, как я должен расстроиться, обновляю страничку, вижу, что здесь уже 50 процентов минус 720.  
  
# Ситуация | Image | Сайт | Cs | Рекомендовать | Видеть | Header | Писать |  
То есть все считается автоматически, он взял одно значение и может с ним отлично работать. Ну что, давайте немножко добавим стилистики. Мне кажется, не хватает контента и немного рассуждений. Так, первое, что меня интересует, это, конечно же, шрифт. Давайте меня интересует это конечно же шрифт давайте с него и начнем шрифт немного отличается плюс не хватает какого-то центрирования и немного значение высоты так я беру копирую название шрифта перехожу в поисковик ввожу fonts google.com переходим на официальный сайт Google. Будем в поиске. Уверен, что эту часть вы выполняете идеально. Выбираем нужное значение. И здесь, соответственно, там подсказки какие есть. Нас интересует. Я видел значение 300. Выбираем. Так, давайте посмотрим, нет ли у меня ничего здесь. Нет ничего, хорошо. Значение 400 видел. И видел значение 700. Если вдруг потребуется 500, 600 или что-то еще мы сможем добавить я копирую данную строчку и добавляю в html я чаще все добавляю где-то в верхней части опять же чтобы ничего не перепутать ну и конечно значение css то есть для всего нашего сами сайта я пишу body и выставляю вот такую вот особенность. Ну что, давайте проверим. Я вижу, что О действительно применилось, действительно работает. Ну, максимум не похоже в макете. Давайте вернем наш родименький цвет. Он хорошенький, поэтому возвращаем на законное место. Так, хорошо. Наша шапка сайта совсем не похожа. И я покажу, как бы я ее сделал. Ну, во-первых, я вижу левую и правой части во вторых я вижу иконочки элементов внутри и естественно у этих иконочек мне главное вот эту пятерку не добавлять то есть это у нас псевдоэлемент явный и туда будет отдаваться значение уже по джава скрипту в итоге хедер left a head сворачиваем не он не нравится пишу хедер left хед сворачиваем, мне он не нравится. Пишу header left, header right и контент. Контент иконки, опять же, может быть любым. Ну, например, вот я сейчас беру эту картинку, перехожу в дизайн, выбираю экспорт, выбираю SVG, соответственно, скачиваю себе. Выбираю экспорт, выбираю SVG, соответственно, скачиваю себе. Как обычно, у меня должна быть папка AMG. И в эту папку я перетягиваю эту картиночку. Можно ли SVG-картинки добавить как-то по-другому? Можно. То есть в этом нет ничего плохого. Давайте, соответственно, выберем нужный нам элементик. Ух ты, даже выше маханули. Весь лого. Так, группа 1, фрейм 2. И прямо при нажатии на него правой клавишей я могу скопировать SVG. \*Copy SVG\*. Ну и давайте тоже посмотрим, как это будет выглядеть. Внутри header left у нас есть лого. А лого ведет он, естественно, у нас на индекс HTML. Так, чуть-чуть двигаю, чтобы было наглядно. И внутри логотипа у нас есть картинка. В папке Image есть group2.svg. Мне не нравится. Я настоятельно рекомендую переименовывать файлы. То есть потом вы не поймете, что за элементы. Естественно, не захочется их пересматривать. Лого. И рядышком с этим лого у нас располагается SVG-изображение. Да, оно такое большое. Но, внимание, мы эту стилистику, если что, можем менять. Плюс ко всему, я рекомендую сворачивать эти блоки. Все. Ну и точно так же нужно будет добавить элементы в header-right. В данной ситуации для нас это не принципиально одинаковое позиционирование и там, и там, поэтому я сделаю только левую часть. Обновляю. Есть два этих блока. Ну и давайте стилистику. Первое, что я, соответственно, вижу, это значение 2, 2, 2. Внимание! Только я вижу значение цвета, я проверяю все остальные. Давайте я щелкну на этот текстовый блок. Черный. Чисто черный. Хорошо. Щелкаю здесь. Какие-то 6F. Понял. Я Я ищу есть ли совпадение. 0, понял. Значение черное. Так, 222, ну там значение 4, понятное дело, что дизайнерская погрешность. И значение в подвале, тоже значение двоечек. В итоге я хочу написать переменную придумываю название и здесь back как background подразумеваю и здесь dark но внимание я не рекомендую привязываться к значению цвета потому что возможно у нас потом поменяются значения так что в данной ситуации здесь лучше для себя определиться. То есть это у нас какие-то background для блоков. А потом определим какие. Хорошо. Значение решеточка 2, 2, 2.  
  
# Смотреть | Color | Видеть |  
Итак, создал переменную, понимаю, что с ней нужно будет работать. Прикольно. И давайте ее сразу же для background определим. Здесь нет рамки. Здесь нет высоты. Здесь есть \*background color\*. И вот она у нас, $back. Если не подсказывает, шелкаем вправо-влево, пишем снова $back, блок. Есть такое. Так, что у нас еще у этого блока? Инспектирование, только значение фон. Можем ли мы здесь задать значение высоты? Вообще здесь уже предусмотрен гамбургер, поэтому в целом я не вижу в этом проблемы. Можно использовать паддинги. Давайте смотреть. Здесь у нас 18 сверху, 18 снизу. 18 сверху, 20 снизу. Да, можно использовать паддинги. Если иконочки стали больше, то можно их переиспользовать. Поэтому смотрите самостоятельно.  
  
# Сайт | Видеть |  
И тот, и другой вариант будет работать. Давайте что-то такое среднее найдем. Я предлагаю прям по 20 пикселей писать. Значение лучше всего выставлять кратное 4. Поэтому 16 мало, 20 оптимально. Паддинг ботом 20 пикселей. Ну что, проверяем. Вот она шапка нашего сами сайта. И естественно центр работает. Но центр у нас не настоящий. Центр не настоящий. Возвращаю 1140. обновляю страничку, вижу что соответственно все есть. Ну и максимум это нужно задать между ними расстояние 41 пиксель. Я вижу что они у нас горизонтально, вижу что центрируются по вертикали и соответственно расстояние 41 пиксель.  
  
# Content | Flex | Смотреть | Видеть | Писать |  
Поэтому для header-left обязательно пишу все рядышком.едера left дисплей \*flex\* line а этом центр и соответственно г 41 пиксель текст проверяем красота ну что теперь я хочу подсказать такие вещи но вот будем считать у нас не самый идеальный пример но опять же невозможно итак давайте поглядим если у меня есть хедер райт мы не наполняем значит полностью хедер готов у нас есть блок который мы назвали top head я добавляю центр естественно нужно чтобы было центрирование все супер я сейчас смогу выставить все эти блоки вот оно у нас значение фона копирую и вижу что такого значения цвета я вообще в принципе нигде не встречал но сейчас продемонстрирую как можно работать с теми же самым значением цвета так топ хэд топ хэд выставляю такое значение и ну вот здесь очень похож на знач высоты, потому что даже если что-то перестроится, оно здесь поместится. Но снова же смотрите по отступам. То есть здесь 59, 60, здесь 65, 66. Ну, в целом я бы выставил 60 пикселей сверху, 60 пикселей сверху, 60 пикселей снизу. Впадинг ботом 60. Что я успел заметить? Что одни элементы слева, вторые элементы справа. Поэтому Display, Flex, Align Items Center, Justify \*Content\*, Space Between. Space Between расстояние между.  
  
# Content |  
Итак, центрирование по вертикали, потому что ты видел, что у нас хлебные крошки чуть-чуть, у нас поменьше по высоте. И разбросали их в разные стороны. Justify Content, Space Between. Так, проверяем. Все убежали. Все супер.  
  
# Видеть | Переменный | Знать |  
Итак, получается, что я сейчас вижу значение фона. Он у нас такой светло-розовый. В целом в припроцессорах можно даже вычитать и добавлять какие-то значения. Тут, опять же, потребуется экранирование. Это достаточно сложная составляющая. Поэтому я просто рассказываю, что так возможно. Почему мы какие-то части, ну, будем считать, опускаем? Вообще, на самом деле, вам самое главное вдохновиться припроцессорами. Очень удобно, очень классно. Мы уже с вами используем, ну, будем считать, опускаем. Вообще, на самом деле, вам самое главное вдохновиться при процессорами. Очень удобно, очень классно. Мы уже с вами используем, ну, будем считать, мощные такие составляющие, которые хочется использовать. Но давайте будем честными, что ну, не хочется усложнить себе задачу, хочется упростить. Поэтому при процессоре или их там идеальное изучение будет, если вы начнете с простенького, потом добавите какой-то функционал, потом выйдите на какой-то продвинутый уровень. Это как будто, опять же, похоже на машинку. Вы купили новый автомобиль, он супер навороченный, он там, не знаю, может держаться в полосе, у него есть какие-то там адаптивные фары, у него есть автоматическая парковка. Нужно ли этим пользоваться, там, не знаю, сразу? Выехал из автосалона, ну-ка давай запаркуйся. Может быть, ты что-то перепутал, может быть, ты еще не освоился, может быть, ты нервничаешь. Да не надо пользоваться постоянно автопарковкой, не надо ездить постоянно на адаптивных фары, еду где-то на трассе, думаю, о, как же удобно. Поэтому здесь точно такая же составляющая. Начните с простого, начните с переменных, начните с математических частей. Вот мы сейчас использовали с вами деление. Все, естественно, будет получаться, и так будет намного проще. Ну что, мы рассмотрели первые два пункта. Это переменные и математические операторы.  
  
# Писать | Product | Видеть |  
Математические операторы затронули, ну, не так объемно, но мы понимаем, что можно складывать, вычитать, умножатьать и делить просто наверняка возникает вопрос а где я буду это использовать где у меня используется та же самая математика поэтому давайте я сейчас создам блок товара мы будем читать опять же также больше обзор на но одновременно с этим мы значение ширины высчитаем из переменной и посмотрим как это работает итак возвращаюсь к нашему с вами коду. Перехожу сюда и буду разрабатывать ProductBox, то есть пока без сортировки. Сворачиваем TopHead. Пишу \*Product\* ProductBox коробочка. И она у нас тоже должна быть с центрированием. И здесь у нас продукт. Ну что, внутри продукта очень много всего. Первое, я вижу вот такую замечательную картиночку. Давайте мы ее скачаем. Экспортирование. Прямоугольное изображение, поэтому выбираю JPEG. Так, папка настроена. Сюда перетягиваю, переименовываю продукт. Ну, в идеале один.  
  
# Смотреть | Image | Product |  
То есть, если у меня их несколько, а у меня их несколько, то будет два, три и так далее. Хорошо, есть Product Image. Потом коробочка. Вот она, смотрите, коробка с контентом. Внутри продукта коробка с контентом.  
  
# Description | Ситуация | Image | Переменный | Cs | Content | Рекомендовать | Вложенность | Смотреть | Link | Видеть | Product |  
Итак, Image. Щелкаю, соответственно, вправо-влево. Подсвечивает мне продукт JPEG. Product. Пока мы дополняем HTML, больше места для HTML. Product. Product. Content. И внутри нашего продукта контента у нас есть название товара. Я настоятельно рекомендую. Сразу же рассуждайте. Вот как бы вы сделали, на сайте перешли, увидели товар. Как перейти на товар? Нажимаем на название товара я настоятельно рекомендую сразу же рассуждать вот как бы вы сделали там на сайте вот пришли увидели товар как перейти на товар нажимаем на название получается что эта ссылка и так а воду можно так и сделать опять же link или нейм что было интуитивно понятно переходов у нас пока никуда нет текст соответственно сюда добавляю дальше у нас есть описание \*description\* или текст и прайс цена у меня это будет два параграфа продукт текст опять же контент любой и здесь продукт продукт прайс значение цены копирую добавляю что здесь хочется отметить понятное дело что если мы говорим про там наполнение она может меняться там про параграфы здесь мы не придираемся если что нам опять же там подскажет самая главная проблема которая сейчас вот вижу понимаю что у меня есть продукт много привязанных элементов давайте даже для картиночки заднем есть продукт, много привязанных элементов. Давайте даже для картиночки зададим класс Product. Image. Все они здесь компактные. И вы заметили, что я большую часть элементов сворачиваю. Вот смотрите, свернул продукты, классно. Развернул продукты, идеально. Но вот в CSS, в данной ситуации в S-CSS, я могу сделать, ну, будем считать, не самое удачное. Я могу написать Product. Могу написать для него стили.е у нас для продукта есть продукт подскажи что ты знаешь так только цвет фона это это для нашего блока ну в данной ситуации меня интересует значение ширины 360 пикселей давайте так и задам ширина 360 пиксель и проблемы начинаются когда вы запустили барабан. Крутим барабан и пишем стилистику, например, для продукт-контента. Вот для продукт-контента почему-то пишут здесь. И тот же самый цвет фона, который у нас здесь появился. Вот он у нас красивенький. Добавляют уже сюда. Что плохо? А что здесь плохого? Ну, давайте поглядим Обновляю страничку Появился товар Появилось то же самое значение Но, внимание! У нас сейчас где-то там стили для продукта Где-то внизу стили для продукта контента Ну, это очень плохо Как же быть? Что делать? Наш с вами вопрос, который решает три процессора Смотрите, как они классно придумали. Мы можем стили, которые относятся к продукту, размещать внутри продукта. Это же гениально! То есть теперь я точно так же, как в CSS, смогу свернуть элементы. Теперь они всегда располагаются рядышком, тебе удобно менять, тебе удобно искать, тебе удобно, опять же, сворачивать, потому что все стили для продукта будут смотреться хорошо. Так, давайте проверим. Стили одного и второго элемента работают, отличий у нас нет, и эта штука у нас называется вложенностью. Поэтому самый главный костяк, самая важная часть, которая у нас есть в уроке, это переменные, математика и вложенность.  
  
# Ситуация | Content | Cs | Flex | Смотреть | Gap | Видеть | Name | Писать | Price | Product | Color |  
Если вы начнете использовать три эти составляющие, знайте, что вы начнете работать с припроцессами постоянно, знайте, что вы сможете развиваться в данной ситуации. И знайте, соответственно, что это не так сложно. Да, дальше будут какие-то такие заковыристые вещи, но этого уже более чем достаточно. Хорошо, мне нравится внешний вид, но в данной ситуации чуть такое у нас какая-то точка с запятой, удалили лишнюю. Сейчас я для демонстрации открою файл стилей. Внимание, я в нем ничего не меняю. Я просто для демонстрации, я хочу посмотреть, а что происходит после того, как генерируется вот этот код CSS. И просматриваю, вижу, что все в порядке. Отдельно продукт, отдельно продукт контент, то есть нет никакого вложенности. Но вот это вот дублирование, или это называется вложенностью, это плохо. Я не хочу, чтобы он писал продукт-продукт-контент. Вот как мне ему объяснить, что нужно сейчас не переиспользовать это слово? И есть такая составляющая. Смотрите, я его выделяю. Удаляю. И пишу значок амперсанта. Знак амперсанта говорит, возьми название селектора. То есть вместо того чтобы написать продукт продукт контент смотрите что он генерирует продукт контент вуаля то есть у меня есть продукт контент продукт текст прайс там имидж все они будут располагаться внутри супер удобно и у всех есть значок амперсанта файл стиля я открывал для демонстрации. В нем никогда ничего не меняем. Его вообще в принципе открывать не нужно. Только чтобы мы увидели ошибку и все. Хорошо. Ну и естественно давайте добавим чуть больше стилистики, чтобы у нас получился тот самый товар. Ну во первых я вижу здесь отступы. Так сверху 25 с чем-то там, 28, 16, 20. Естественно, если у нас есть подобного рода проблемы, я бегу к дизайнеру, говорю, слушай, дружище, ну что-то не так. Плюс ко всему, если у нас есть дробные значения, их нужно исправить, и отступы справа и слева должны быть симметричные. Как симметричные? Я здесь четко вижу там какое-то 28 значение. Туда просто слово не поместилось. Вот слово не помещается, дизайнер не растягивает этот блок, и получается из-за того, что слово прыгнула на новую строчку отступ справа стал немножко другим и того отток сверху и снизу симметричные у меня будут по давайте падинг пишу общий сразу сверху и снизу у нас 24 там у нас давайте чтобы у нас точно 25 и 20 давайте 24 до 24 пикселя а слева и справа по 16 пикселей то есть для блока контент я соответственно добавил отступы все хорошо стилизуем каждый из блоков здесь нет никакой загадки просто нажимаю просто копирую но и кстати черное значение цвета вообще можно тоже вынести то есть это значение текста. $color text. То есть для большинства текста я увидел, что значение прям черненькое, поэтому я его сюда и добавляю. txtx у нас есть продукт ampersand 2 нижних подчеркивания heading или как я его назвал \*name\*. name Для product name выставил такие параметры И черное значение цвета, естественно, меняю на переменную Dollar color text Классно Здесь есть отступы 12, что-то там 16 Да, все верно Давайте их просто значением gap зададим То есть для \*product content\* я пишу \*Display Flex\*. Хорошо. \*Flex Direction\*. Flex расположение колонкой. И я увидел только что отступ между ними 16 пикселей. Так. И теперь стилистика для нашего прайса. Вот оно опять же розовенькое значение цвета. Мы тоже укажем. Ampersand. \*Price\*. И здесь значение. Ну что. Есть у нас такой подход обновляю ух ты какой это большой отступ давайте проверять мы не обнулили отступы в принципе и так звездочка самое главное то я не написал марджин 0 по 1 к 0 обновляемю. Смотрю. О, теперь 1 в 1, как в макете. Значение есть. Что-то он какой-то слишком у нас крутой. Тэк-тэк-тэк-тэк-тэк. Это у нас значение light для нашего блока. Вот смотрите. Я опять же понимаю, что все значения здесь. То есть я ни в коем случае не перепутаю. А текст я вообще не стилизовал. Вот так-то. Значение текста и стилистика для текста цвет какой-то не совпал он другой и мы выставляем его сюда ну что теперь точно похоже полностью готов у нас с вами товар и там и там я например использовал перемен но где математика математика начинается на вымч, с самого простого. Я понимаю, что отступ между блоками у нас 30. И давайте посчитаем. Есть общее значение ширины. Раз. У нас есть три наших блока. Два. У нас есть расстояние между этими блоками. Три. Получается, что если я из общего расстояния вычту двойной отступ. Давайте, кстати, отступ тоже запишем в переменную.  
  
# Сайт | Переменный | Gap | Видеть | Писать | Знать |  
Итак, я пишу $GAPGRID. GAP – это отступ, а отступ тоже запишем переменную и так я пишу доллар гэп а grid гэп это отступ grid сетка вот отступу сетки я вижу стандартные 30 30 пикселей и давайте прямо здесь математику тогда развернем я сейчас могу создать доллар треть вот я не знаю например как пишется треть переводчик пишу треть только с опять же треть сайта сайта треть нашего сами сайта что мы делаем мы берем ширину сайта то есть внутри переменной я могу переиспользовать переменные. Это все в скобочках, то есть ширина сайта. Минус, здесь двойной гэп, доллар, гэп, грит. И эту штуку разделить на 3.  
  
# Сайт | Переменный | Flex | Рекомендовать | Вложенность | Видеть |  
Итак, треть сайта теперь зависит от отступа и теперь зависит от ширины Ну и давайте это все добавим в продукт Давайте обновляем Ух ты, немножко поменялось, сейчас посмотрим Может быть у нас математика хромает Что говорит? 370 получилось Посмотрим, все ли верно мы посчитали. 360, 360, 360. А все вместе 1140. Где-то мы 10 пикселей потеряли. Что сейчас удобно? Я, если вдруг меняю 1440, то есть представляете, да? Обновляю, вижу, что для продукта у нас применилось значение 470. То есть ширина сайта высчитывает всегда треть ну кстати для картинки тоже было бы здорово наша ampersand и мэйдж здесь лучше написать ширину 100 процентов и плюс видимо дисплей flex нужен для нашего продукта дисплей flex flex direction колом то есть чтобы не было отступа видите меня чуть смущает этого белая линия и от нее хочу display flex flex direction column. То есть, чтобы не было отступа. Видите, меня что-то смущает эта белая линия. Я от нее хочу отказаться. Все, видите, она исчезла. Так, возвращаю. Возвращаю все, как было. Это математическая задачка. Разберитесь, где мы 10 пикселей потеряли. Все, продукт готов. Математика, переменные, вложенность. И как только мы завершили работу с продуктом, все, я его просто сворачиваю. Представьте себе, насколько будет наглядный код. То есть все, что завязано с продуктами, у нас находится здесь. Я быстренько это сверну, здесь сверну, и там, и там выглядит красиво. Поэтому настоятельно рекомендую к использованию. Ну что, мы разобрали 90% от того, что нам с вами необходимо. Оставшиеся 10% они больше ситуативные. То есть, например, в нашем проекте это мы не так часто встретим. Возможно, я даже не рекомендую что-то использовать в виде примесей, например. Но разобрать мы их должны. Опять же, эту часть очень сложно запомнить, потому что и переменные сразу же знаешь, где применить и как. И ту же самую математику она встречается опять же везде.  
  
# Переменный | Cs | Рекомендовать | Вложенность | Импорт | Sa | Видеть | Знать | Sc |  
Вложенность там бесконечная и так далее. Но вот если мы говорим про какие-то примеси, то это уже более такой абстрактный пример. Ну и поэтому давайте более детально разбираться. Если вам они потребовались через полгода, что я рекомендую? Вы переходите на сайт. То есть опять же там VELSAS. Вот я вижу переменные, вот у нас есть вложенность. Сейчас будем еще \*импорты\* смотреть, отдельные модули. Кстати, давайте, чтобы у нас названия совпадали, вернем значение. Где он? Вот здесь, по-моему, мы смотрели. SAS. Да, то есть документация на русском, чтобы мы, опять же, не потерялись. Точно так же установка и основы SAS. Первая при процессоре. Вот они, переменные. Вот она у нас, выложенность. Фрагментирование. Фрагментирование. Ну и я предлагаю с нее и начать. Честно говоря, на начальном этапе можно и нужно разбивать файлы. И наверняка у вас такая мысль была. Она чаще всего встречается, ну, будем считать, в двух самых таких глобальных категориях. Во-первых, это когда у тебя есть интернет-магазин и есть вторая страница. И ты такой, а где размещать эти стили? Где-то внизу, где-то вверху, где-то отдельно. Получается, что в идеале это, чтобы у вас был один общий файл стилей. И можно разбивать по категориям. Можно отдельно стили для хедера. Можно отдельно стили для страницы каталога. Можно отдельно стили для кнопки вынести, если хотите. Ну вот как только ученики это узнают, они такие, так это же гениально. Заходишь, миллион стилей. Просто этих папок, там не знаю, файлов у них очень много. Друзья, сразу договариваемся. Зачем вам это? То есть зачем вам отдельные файл стилей для кнопки? Зачем вам это нужно? Поэтому отметаем эту идею. Для отдельной страницы можно, для медиазапроса можно, для вот таких вот мини-элементов не рекомендуется. Что здесь, соответственно, пишут? Самое важное, когда вы создаете новый файл, это добавить нижнее подчеркивание названия файла SCSS. Нижнее подчеркивание нужно для того, чтобы у нас на выходе был один CSS. Если мы не используем нижнее подчеркивание, то из этого припроцессорного файла тоже появится CSS. Их уже будет 2, а потом их будет 5. То есть на выходе всегда 1 CSS файл. Припроцессорных много. CSS 1.  
  
# Рекомендовать | Переменный | Cs |  
Итак, что мне хочется? Я рекомендую переменные вынести в отдельный файл. Но, казалось бы, зачем? Смотрим. Ctrl-N, Ctrl-S. Нижнее подчеркивание. vars. Как переменный. S CSS. Нуижнее подчеркивание. vars. vars как переменный. s css. Ну и, соответственно, сохранить.  
  
# Писать | Переменный | Cs | Расширение |  
Итак, у меня есть отдельный файл с переменными. Из него не появляется css. И здесь все переменные. takes. Все переменные. Где вы? Копирую. Добавляю. Здесь вместо этих переменных я пишу собачка import. Внимание, в кавычках vars. Без нижнего подчеркивания, без расширения файла. Ну и, соответственно, точка с запятой обязательно. Зачем мы это сделали? Файл vars я могу сместить вниз. Вниз. Вот он. Опять же, чаще всего больше разрешение экрана. То есть, как я чаще всего занимаюсь кодингом? Мне нужно больше разрешения экрана. Вот у меня переменные здесь, HTML здесь, CSS здесь. Понятное дело, что для демонстрации я делаю разрешение побольше.  
  
# Переменный | Рекомендовать | Смотреть | Знать | Sc | Color |  
Но когда вы будете работать, вынесены отдельные переменные. Они у меня всегда под рукой. Я всегда знаю, как они называются. Я всегда могу сюда добавить новые. Ну и, естественно, мне не нужно, опять же, крутить барабан, крутим вверх-вниз, исправлять все эти значения. Давайте посмотрим, все ли у нас работает, ничего ли не поменялось. Так, приближаю обратно, чтобы вам было комфортно. Обновляю страничку. Ничего не поменялось. И теперь давайте в файле. Вот у нас отдельный файл стилей. Круто. В этом файлеайле я опять же меняю на значение фиолетового цвета потемнее будет а вот так смотрите этого здесь здесь эту часть кстати забыл прикольно внутри продукта удобно мы ищем где же наше значение прайс цена и меняем на переменную color select сайт есть такое обновил вот она у нас появилась и так можно создавать отдельные файлы можно их подключать первое файл создаем через нижнее подчеркивание дальше расширение с с с подключаем его через импорт без нижнего подчеркивания, без SCSS. Это самое важное. Ну что, круто, можно приплюсовывать, главное не очень много. Бежим еще дальше, хочу еще что-то вкусненькое, мне прям понравилось. Импорт мы рассмотрели, то есть есть подключение, и, соответственно, бежим к миксинам. К миксинам я не рекомендую обращаться на таком глобальном этапе, честно говоря, они еще здесь, знаете, так написаны, ты такой думаешь, что происходит?  
  
# Mixins | Писать | Flex |  
Как с этим работать? Ну, давайте глобально. Миксины лучше всего хранить в отдельном файле. Поэтому мы с вами так и поступим. Я нажимаю Ctrl-N, Ctrl-S. Нижнее подчеркивание \*mixins\*.scss. И сюда мы можем добавить, что хотим. Так, я скопировал их пример. Но мне, если честно, он не очень нравится. Что было бы комфортно? Ну, например, я создаю Mixin. Что это означает? Это означает как будто функция. Зачем мне нужна функция? Функция собирает в себе несколько действий. Я здесь пишу центр flex. Проперти я могу сюда передать переменную, пока не будем созданием простую. И здесь мы группируем все, что хотим. Display.Flex. Justify.Content.Center. Align.Items.Center. Flex.Wrap.Wrap. То есть вот такую штуку я встречаю везде и всюду. Она, будем считать, очень популярная. Я ее сгруппировал и назвал CenterFlex.  
  
# Ситуация | Смотреть |  
Получается, что мне... Смотрите, Include include у них какой-то странный здесь include а трансформация просто идет я понял include трансформ и значение эту штуку мы удаляем итак mixin это отдельный в данной ситуации файл это не обязательно можно создавать его в нашем файле я его подключил вот здесь вот собачка импорт в кавычках mixin mixin и отлично и теперь если мне нужно что-то центрировать где мы это добавим давайте давайте давайте ну продукты некрасиво, у контента. Смотрите, так часто встречается, что мы даже нигде не увидели. Для продукта, продукт бокса. Прикольно. Мне кажется, это интересное решение.  
  
# Flex | Рекомендовать | Писать | Product | Box |  
Итак, продукт бокс. Я просто пишу include. И здесь название моего миксина. Я написал центр, flex и круглые скобочки. В итоге для продукт бокса вместо четырех строчек, это центрирование, горизонтальное, вертикальное, перенос строки, но сам дисплей флекс, я задаю всего одну. Include, центр, флекс. Давайте поглядим. Обновляю страничку, все сработало. То есть для Product \*Box\* у нас сейчас все четыре стиля потянулись. Что это означает? Я могу сгруппировать какие-то части и, соответственно, их просто переиспользовать. Конечно, на начальном этапе у тебя просто голова кругом. Ты думаешь, батюшки мои, одно, второе, там где-то как-то собирается. Миксины не рекомендую. Поэтому да, к ним можно будет прийти, но не на старте. Так, дальше у нас идет наследование.  
  
# Писать | Flex |  
Похожая составляющие и чаще всего бывает в каких-то вот будем читать у нас есть такой примерчик вот у нас есть элементы слева есть элементы справа что мы здесь увидели дисплей flex и line items центр и значение г получается что вот такие штуки можно перезаписать и Итак, я пишу знак процентов message shared. И здесь какое-то значение. Чтобы переиспользовать его, я пишу собачка extend и message shared. Что это означает? Что внутри нашего месседжа через запятую пропишется данный блок. Ну и, конечно, давайте что-нибудь такое создадим.  
  
# Ситуация | Cs | Flex | Рекомендовать | Смотреть | Header | Знать |  
 тренд он там flex то есть я придумываю название через значок есть такое дальше но дисплей flex например и flex wrap wrap самые такие базовые вещи и вот эту штуку я хочу добавить ну например для кого Для нашего блока header. Собачка extend. Вот она, extend. И наш значочек. Доллар. Не доллар, а проценты. Flex. Эту штуку мы сможем увидеть в файле CSS. Опять же, чтобы не потеряться. Я его перетягиваю сюда. В нем ничего не не меняю он не нужен только для демонстрации смотрите отдельно у хедера написано дисплей flex flex wrap wrap то есть наверняка у вас была такая история что вы хотели написать через запятую поэтому можно будет писать собачка extend процент flex смотрим через запятую хедер left хедер записан то есть если вы думаете приплюсовать что-то через запятую тогда это идеальный способ метод и я настоятельно рекомендую к использованию на начальном этапе опять же это следующий шарочек знаете как игра то есть только ты освоил первые составляющие переходи ко вторым не хватай за вторые, потому что они чаще всего сбивают. Ну и в итоге математические операторы. Что мы можем заметить? То есть в данной ситуации при процессоре ничего не сломали. То есть это просто апгрейд нашего CSS.  
  
# Заключение  
И смотрите, насколько удобно будет в дальнейшем с ними работать. Редактирование идеальное. Опять же, математика все отдельно высчитывает. Перемены все в отдельности запоминают Вложенность придает нам, соответственно, наглядность Фрагментирование можно разбить на отдельные файлы Это будет удобно, если у нас вынесен, например, медиазапрос 1, медиазапрос 2 Ну и для отдельных файлов Я настоятельно рекомендую импорты подключать выше То есть в идеале, чтобы у нас был файл с импортами А потом уже разбиение на другие элементы Примеси. Ну, мне кажется, прикольно, но опять же ситуативно. То же самое наследование. Ситуативно. Поэтому в данной ситуации мы идем по шагам. Я надеюсь, мне удалось вдохновить. И поэтому давайте опять же поговорим про следующий препроцессор. Препроцессор лес в данной ситуации ничем не будет отличаться. Это как конкурент, но внимание, которое потеряло актуальность. Это знаете, что-то в духе у нас были дискеты. Я прям пользовался дискетами, мне казалось норму. А потом я вижу, что уже используются диски. Чем отличаются? Лучше, функциональнее, потом про них забыли. То есть уже покупаешь себе системный блок, а там уже нет. И такой, как же так без дискет? На самом деле нормально. Вот лес то же самое. То есть чуть-чуть по-другому создаются переменные. Та же самая математика, там та же самая вложенность. Нет, например, циклов. Мы сегодня не рассмотрели, как я говорил, еще у нас будет. Простыми словами, если мы рассматриваем другие припроцессоры, это нормально. Если вы, не знаю, пришли в компанию, они используют припроцессоры лес, отлично. То есть разница будет \*минималистичная\*. Вам нужно будет, не знаю, час потратить времени, чтобы перей они используют припроцессор LES, отлично. То есть разница будет минималистичная, вам нужно будет, не знаю, час потратить времени, чтобы перейти на новый припроцессор. Вы можете его использовать в команде. Плюс ко всему, опять же, я в текстовом варианте прописал эти блоки, так что здесь переживать точно не стоит. Ну что, давайте подойдем к итогу нашему сегодняшнему занятию. Надеюсь, вам очень понравился, опять же, наш с вами урок по простому принципу. Я, во-первых, рассказал вдохновение про курс. Круто. Потом мы освоили теоретический вариант. Я, опять же, подсказал все-все-все базовые вещи. Естественно, подсказал по установке припроцессора. Мы с вами выбрали самый популярный и сразу же разбирали его на практике. То есть мы рассмотрели припроцессор, мы рассмотрели весь необходимый функционал, как все это устанавливается, как все работает. Мы рассмотрели, что важно, что менее приоритетно, ну и, естественно, смогли сравнить тот же самый припроцессор SAS с припроцессором \*LESS\*. То есть, по факту, разница будет минималистична, отличие там, не знаю, всего часик где-то посидеть. А у меня на сегодня все. Поздравляю с прохождением урока. До встречи в следующем. you