**Инструкция!**

Для работы Фреймворка в самом начале всех файлов должно быть прописано **@use "sass:math"**

По умолчанию прописано в файле сборки style.sass.

**1 Подключите style.sass к вашему проекту с сохранением путей**(если нужно переписать – делайте вручную).

2 Установите сниппеты.

**Сниппеты Фреймворка:**  
  
Сниппеты из папки --> snippets

Поместить в папку “snippets” VScode  
путь C:\Users\..”имя пользователя”… \AppData\Roaming\Code\User\snippets  
пример: “C:\Users\werw\AppData\Roaming\Code\User\snippets”  
  
Онлайн редактор сниппетов: <https://snippet-generator.app/>

Так же путь через VScode:

**File** > **Preferences** (**Code** > **Prefere nces or Code > Settings on macOS)**

**Файл** > **Настройки** > **Настроить пользовательские фрагменты кода** > **Новый файл(см внизу)** > **ввести имя (создать)** > **клик по вкладке сниппета** > **открыть в проводнике (попадем в папку сниппетов – создайте ярлык для удобства)**

**Инструменты:  
 Технические классы:**

**Основные ТК расположены в папке “general-classes”.**

**Синтаксис:**

**.\_class-name сокращение от именования: .page\_\_class-name   
 Что соответствует именованию по БЭМ.**

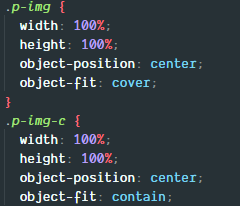
**Краткий список:**

**.\_container - базовый контейнер**

**.p-img - для img – устанавливает**

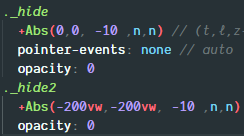
**картинку в размер родителя**

**.p-img - (object-fit: contain)**

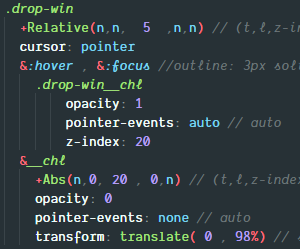
****

**.svg**

**устанавливает дочерний SVG в размер этого класса**

** .\_hide , .\_hide2**

**Скрывают блок без display: none**

**.drop-win**

**.drop-win\_\_chl - выпадающее окно на ховере**

**.\_ibg - родитель**

**.ibg-img -- изображение   
 превращает картинку в бэграунд изображение**

**.ibg-img-с (object-fit: contain)**

**Функции:**

**Файл - \_\_1-unitconvWERW.sass**

**Функции принимают 2 значения**

**rem($px, $context:16)**

**1 – целевое  
 2 – контекстное –( относительно чего)**

**Можно вводить только 1 значение , второе значенеи функция подставит по умолчанию**

**В фреймворк включены следующие функции:**

**rem(16) = 16px = 1rem**

**переводит px в rem**

**em(16) (если font-size: 16px (rem)) = 16px = 1em**

**em(16,32)** **(если font-size: 32px (rem)) = 16px = 0.5em**

**переводит px в em**

**vw($px, $vw:$maketSize)**

**переводит px в wv – от значения переменной $maketSize(размер макета)**

**width: vw(40) = 40px на экране размера макета = 2.7777777778vw**

**width: vw(40, 780) = 40px на экране размера 780px**

**vh($px, $vH: 1080)**

**переводит px в vh – по умолчанию контекс 1080px (FullHD)**

**pr($target:1200, $context: $containerSize)**

**переводит px в % – по умолчанию $containerSize – размер конгтейнера**

**width: pr(40) === width: 2.6666666667%;**

**cof($target , $context) // coefficient**

**высчитывает коэфициент числа**

**cof(11 , 145) == 0.075862069**

**0.075862069 часть числа 145 которая равна 11**

**fr($target , $context)**

**переводит px % в fr**

**пример :**

**grid: none / fr(450,900) 1fr**

**результат:  
 grid: none/0.5fr 1fr;**

**снипеты функций:  
 r 🡪 rem()**

**em 🡪 em()**

**vw 🡪 vw()**

**cf 🡪 cof()**

**pr 🡪 pr()**

****

**Миксины:**

**В качестве передачи пустого значения используется n (none)**

**+size( n , 20 , n)**

**Не указанные значения по умолчанию передаются пустыми.**

**т.е можно указывать только первые значения**

**+size( 20,12 )**

**В качестве переключения функционала работы миксинов используется проефикс после названия**

**Миксин без префикса – переводит значения в rem**

**Пример: +size( 20)**

**Префикс 3**

**Пример: +size3( 20px ) - Принимает полные значения**

**+size3(rem(20) )**

**Префикс 4**

**Миксин переводит значения в EM ,в певом значении нужно добавить развер шрифта как контекст , значени n – контекста == значению размера шрифта 16**

**Пример: +size4(22, 20, n )**

**Префикс 5**

**Миксин переводит значения в EM . Указывать контекст не нужно – по умолчанию контекст размера шрифта для миксина равен 16**

**Пример: +size4( 20, n )**

**Миксин size описание:**

**+size(➡️,⬇️,🔄,🈯,🐬,⬇️,➡️)**

**Вызывается снипетом sss**

**➡️ width ⬇️ height: 🔄** **border-radius:**

**🈯** **background-color:**

**🐬 overflow: hidden**

**⬇️:not(:last-child) margin-bottom:**

**➡️:not(:last-child) margin-right:**

**По умолчанию ⬇️ height == ➡️ width**

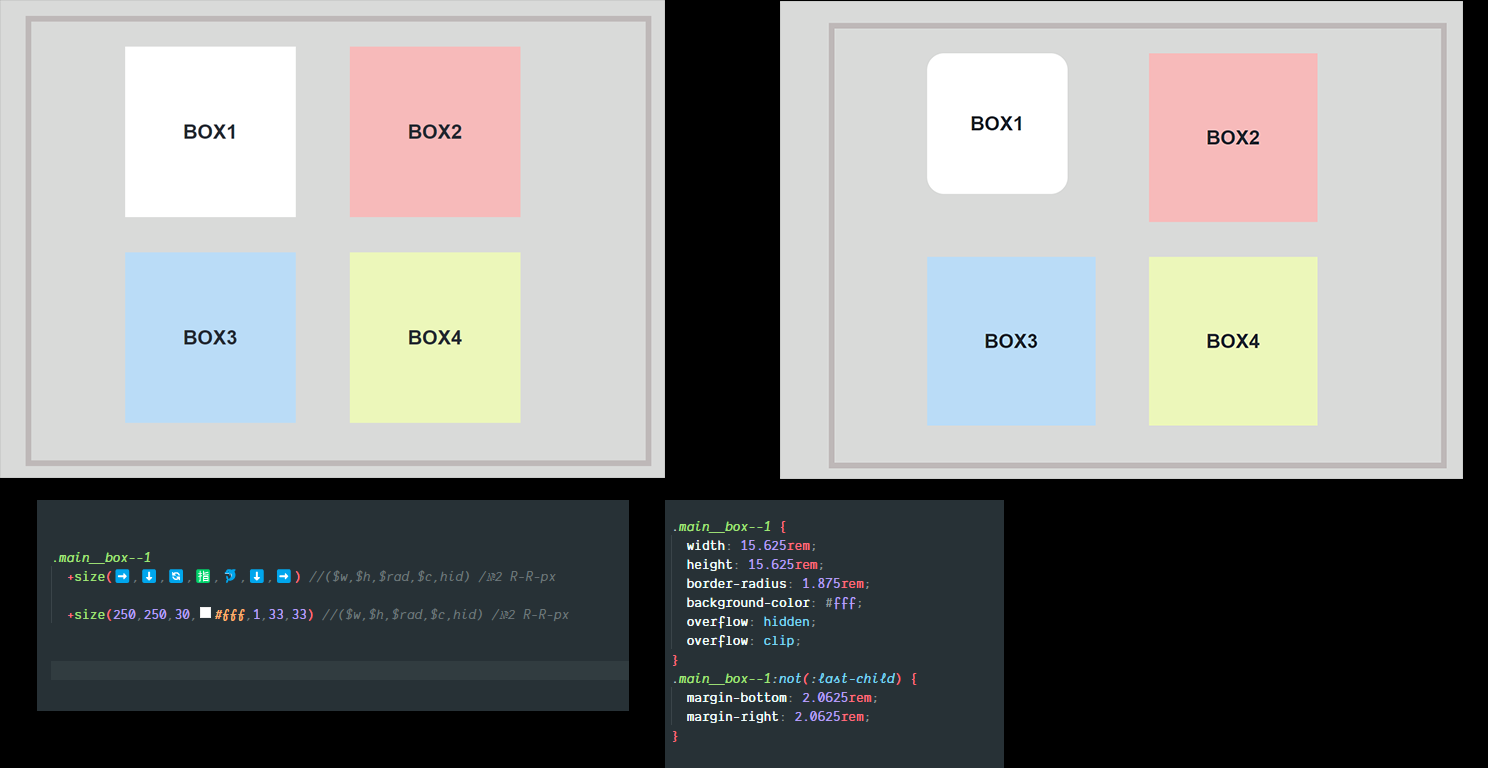
**Если записать +size( 20 ) то height width будут по значению 20**

**+size(20,,w,🈯,🐬,⬇️,➡️)**

**Значение w для ⬇️ height присвает значение ➡️width**

**Значение с для ⬇️ height создаёт круг по размеру ➡️width**

**Значение с для 🔄** **border-radius: 50%**

****

**Миксины позиционирования описание:**

**+Abs(⬆️,⬅️,n,⬇️,➡️) position: absolute**

**+Fix(⬆️,⬅️,n,⬇️,➡️) position: relative**

**+Relative(⬆️,⬅️, n ,⬇️,➡️) position: fixed**

**Вызывается снипетом abs ,fix , rrr**

**ПРЕФИКС 0 отменяет добавление стиля position:**

**+Abs0(⬆️,⬅️,n,⬇️,➡️) position: не будет объявлен**

**⬆️top:**

**⬅️left:**

**N z-index:**

**⬇️ bottom:**

**➡️right:**

**Значения указываются полностью:  
+Abs(rem(10),⬅️,2,⬇️,➡️)**

**Миксин +Font описание:**

**+Font(s,l,w, 🈯, name ,ls,🐬,⬅️)**

**Вызывается снипетом fff**

**ПРЕФИКСЫ: 3,4,5 см. миксин +size()**

**S font-size:**

**L line-height – кроме префикса 3 переводит px в коэфициет  
 если S не указан или указан как n(none) то принимает фактическое значение т.е. 45px +Font(n,45px)**

**W font-weight: может принимать сокращённые значения 1-9 или целевые значения  
 4 == font-weight: 400**

**🈯 color:**

**Name font-family:**

**Ls letter-spacing: кроме префикса 3 переводит px в em**

**🐬 принимает значения 1,2,3**

**1 text-transform: uppercase**

**2 text-transform: capitalize**

**3 text-transform: lowercase**

**⬅️** **text-align: принимает значения 1,2,3**

**1 text-align: left**

**2 или с text-align: center**

**3 text-align: right**

**Миксин +Flex описание:**

**+Flex(1,jc ,ai ,g ,fw ,ac)**

**Вызывается снипетом f**

**1 – первое значение – объявление.**

**1 == display: flex**

**2 == display: inline-flex**

**c , cl , clm == flex-direction: column**

**display: flex**

**2c , 2cl , 2clm == flex-direction: column**

**display: inline-flex**

**2n , nc == flex-direction: column ( не присваивает flex)**

**1n == flex-direction: row ( не присваивает flex)**

**jc justify-content:**

**1 == flex-start**

**2 , c == justify-content: center**

**3 == flex-end**

**sb , 10 == space-between**

**se, 12 == space-evenly**

**sa , 13 == space-around**

**ai align-items:**

**1 flex-start**

**2 center**

**3 flex-end**

**g gap: - передавать целое значение --> +Flex(1,jc ,n ,rem(20))**

**fw - flex-wrap:**

**1 , w == flex-wrap: wrap**

**0, nw == flex-wrap: nowrap**

**3 == flex-wrap: wrap-reverse**

**ac - align-content:**

**1 == flex-start**

**2 , c == center**

**3 == flex-end**

**4 == stretch**

**sb,11 == space-between**

**sa , 12 == space-around**

****

**Миксин +Grid описание:**

**Вызывается снипетом g , gta(полный)**

**+Grid(1, jc ,ac ,jitm ,ai ,gap ,gaf🔜 )**

**grid: none / repeat(2, auto)**

**пример функции fr() - получаем значение где 433px = 1fr(целая доля) ,**

**а 222 переводятся в относительный fr == fr(222,433)**

**grid: none / fr(222,433) 1fr**

**1 – первое значение – объявление.**

**1== display: grid**

**2 или i== display: inline-grid**

**3 или s == display: subgrid**

**jc justify-content:**

**1 == flex-start**

**2 , c == justify-content: center**

**3 == flex-end**

**4 или 5 ==** **stretch**

**sb , 11 == space-between**

**se, 12 == space-evenly**

**sa , 13 == space-around**

**ac align-content:**

**1 == start**

**2 или c == center**

**3 == end**

**4 или 5 == stretch**

**10(sb)== space-between**

**12(se) == space-evenly**

**13(sa) == space-around**

**Jitm justify-items:**

**1 == start**

**2 или c == center**

**3 == end**

**4 или 5 == stretch**

**ai align-items:**

**1 == flex-start**

**2 == center**

**3 == flex-end**

**4 или 5 == stretch**

**Gap gap: - передавать целое значение --> rem(20) или 20px**

**gaf🔜 grid-auto-flow:**

**1 == grid-auto-flow: row**

**2 == grid-auto-flow: column**

**3 == grid-auto-flow: dense**

**Миксин +center описание:**

**Вызывается снипетом ccc**

**Центрирует блок дочернего элемента**

**display: flex**

**justify-content: center**

**align-items: center**

**Миксин +🦄gridElm описание:**

**Вызывается снипетом gi (grid item)**

**+🦄gridElm(n ,n ,n ,n ,n)**

**1 значение:justify-self:**

**1 == start**

**2 == center**

**3 == end**

**4 == stretch**

**2 значение:align-self:**

**1 == start**

**2 == center**

**3 == end**

**4 == stretch**

**3 значение: grid-column:**

**Принимает значение**

**Пример grid-column: 1 / span 2 == +🦄gridElm(n ,n ,** **1 / span 2,n ,n)**

**4 значение: grid-row: Принимает значение аналогично grid-column:**

**5 значение: order: Принимает значение**

**Пример order: 4 == +🦄gridElm(n ,n ,n,n ,4)**

**Миксин +transform описание:**

**Вызывается снипетом tr**

**+transform( ➡️,⬇️, 🔄,🐇,🕐,📣,🛂) (x,y,rot,scl,time,type,dur )**

**➡️значение:**

**transform: translate( ось X)**

**+transform( 20px ,⬇️, 🔄,🐇,🕐,📣,🛂)**

**⬇️значение:**

**transform: translate (ось y )**

**🔄 значение:**

**transform: rotate**

**переводит значение в deg**

**+transform( ➡️ ,⬇️, 90,🐇,🕐,📣,🛂)**

**🐇 значение:**

**transform: scale**

**+transform( ➡️ ,⬇️, 🔄,1.2,🕐,📣,🛂)**

**🕐 значение:**

**transition-duration** — длительность перехода;

**📣 значение:**

**transition-timing-function —** функция, описывающая скорость изменения

свойства;

**🛂 значение:**

**transition-delay** — задержка перед началом изменения.

**🕐 📣 🛂 - передаются фактическим значением**

**+transform( ➡️ ,⬇️, 🔄,🐇,3s,** ease-in-out**,0.5s)**

**Миксин +** **Dynamic описание:**

**Вызывается снипетом d**

**+Dynamic2(1, -1 , 0 , 1200, stop)**

**+Dynamic(f, 0 , 0 , 1200) //t 1:f 2:w 3:h/max/min/wS2**

**+Dynamic:**

**Аналогичен функции clamp() css – но с широкой поддержкой**

**Вызывет контрукцию : max(rem(), min( vw(), rem() ) )**

**1 значение: type (тип)**

**Задаёт название свойства с которым работает миксин**

**1 или f == font-size:**

**2 или w == 'width'**

**3 или h == 'height'**

**pt == 'padding-top'**

**pb == 'padding-bottom'**

**2 значение: максимальный размер**

**3 значение: минимальный размер**

**4 значение: контекст – ширина окна при котором**

**максимальный размер в пикселях равен значению 2.**

**+Dynamic2:**

**+Dynamic2(1, -1 , 0 , 1200, stop)**

**Имеет другую концепцию**

**min( … vw(…) ) или max( … ,vw(..) )**

**1 значение: name аналогично type (тип) +Dynamic**

**2 значение: тип – задаёт направление работы миксина**

**-1 свойство css будет искать минимальное значение min**

**Динамическое значение будет в vw**

**1 свойство css будет искать максимальное значение max**

**-2 аналогично -1 Динамическое значение будет в %**

**2 аналогично 1 Динамическое значение будет в %**

**3 значение: базовый размер**

**4 значение: контекст – ширина окна**

**5 значение: stop**

**Значение при котором тип -1 будет максиальным значением**

**тип 1 будет минимальным.**

**По умолчанию stop равен значению 3**

**пример:**

**+Dynamic2(1, -1 , 55 , 1200, 22)**

**это**

**font-size: min(1.375rem, 4.5833333333vw);**

**аналог в пикселях**

**font-size:** **min(22px,vw(55,1200);**

**т.е font-size будет увеличиватся или уменьшатся в соотношении 55px**

**к размеру окна 1200px но не будет больше 22px**

**Миксины +sizeMax +sizeMin описание:**

**Вызывается снипетом sm**

**Аналогичны миксину +size**

**Включая +sizeMax3 +sizeMax4 +sizeMax5 - префиксы**

**Но в отличии от +size**

**Задают не width height  
 а max-width: max-height:**

**И min-width min-height аналогично**

**Миксин +N-last-child описание:**

**Задаёт отступы и осуществляет поддержку gap для Flex.**

**+N-last-child( 1, ⬇️ , ➡️)**

**Вызывается снипетом nlc**

**1 Первое значение – Тип: $type**

**⬇️ margin-bottom: переводит значение в ремы**

**➡️ margin-right: переводит значение в ремы**

**Пример: +N-last-child( 1, 12 , 33)**

**Значения $type**

**1 – задаёт :not(:last-child)**

**2 – задаёт :not(:last-child)**

**И обнуляет ⬇️➡️ если браузер поддерживает gap для Flex**

**3 –**

**Применяется для Flex блока**

**задаёт отрицательные значения отступа ⬇️➡️**

**Объявляет > \* - задаёт для всех дочерних блоков первого порядка ⬇️➡️**

**4 и 5**

**Применяется для Flex блока**

**Дублирует тип 3**

**обнуляет ⬇️➡️ если браузер поддерживает gap для Flex**

**задаёт gap: ⬇️➡️ если браузер поддерживает gap для Flex**

**7**

**Дублирует тип 4 и 5**

**Но не объявляет gap: ⬇️➡️**

**Т.е gap: можно задать в ручную с другими значениями чем ⬇️➡️**

**+N-last-child4 – аналогичен +N-last-child – но перводит значения не в rem а в em**

**Для +N-last-child4 нужно задавать контекст размера шрифта**

**Пример +N-last-child4(20, 1, ⬇️ , ➡️)**

**Значение n для конткста = 16 Пример**

**+N-last-child(n, 1, ⬇️ , ➡️) == +N-last-child(16, 1, ⬇️ , ➡️)**

**+N-last-child5 – аналогичен +N-last-child5 – контекст по умолчанию равен 16, задавать не нужно Пример +N-last-child5( 1, ⬇️ , ➡️)**

**Снипет absc описание:**

**+Abs(50%,50%, n ,n,n)**

**transform: translate(-50%, -50%)**

**Выравнивет блок при абсолютном позиционировании по центру**