WEB Javascript

# **WEB Javascript**

**Sublime text 3, 4**

**(лучше для вёрстки)**

**VS Code**

|  |  |
| --- | --- |
| **Ctrl + u , F12** | просмотр кода |
| **Shift + Alt + w** | обернуть выделение в Тег |
| **Ctrl + /** | комментирование (раскомментирование) строки |
| **Alt + .** | Закрыть незакрытый тег |
|  |  |
|  |  |

(пример: **ol>li\*10**, Tab - создание списка с 10 элементами

**.container>.header+.content>ul>li\*5{item$}**

**.container>.header>h1{Hello world}+.content>ul>li\*5{item$}**

**#page>div.logo+ul#navigation>li\*5>a{Item $}**

превращается в

**<div id="page">**

**<div class="logo"></div>**

**<ul id="navigation">**

**<li><a href="">Item 1</a></li>**

**<li><a href="">Item 2</a></li>**

**<li><a href="">Item 3</a></li>**

**<li><a href="">Item 4</a></li>**

**<li><a href="">Item 5</a></li>**

**</ul>**

**</div>**

**"color\_scheme": "Packages/Base16 Color Schemes/Themes/base16-ocean.tmTheme",**

**"theme": "auto",**

**"margin": 0,//Убирает отступы**

**"tab\_size": 2,//Размер табуляции**

**"dark\_theme": "Default Dark.sublime-theme",**

**"light\_theme": "Default Dark.sublime-theme",**

**"font\_size": 11,**

**"draw\_white\_space": "all",//Отображает непечатаемые символы**

**"draw\_indent\_guides": false,//Убирает направляющие линии**

**"remeber\_open\_files": true,//Помнит открытые ранее файлы**

**"trim\_trailing\_white\_space\_on\_save": false,//Отключает самодеятельность программы**

**"word\_wrap": false, // запрещает перенос строки**

**user.name=Gitzoteevasd**

**user.email=zoteevasd@gmail.com**

GitHub Token: gho\_\*\*\*\*\*\*\*\*\*\*

GitHub Gist: 4e69952bd5e6a5aa4707f774dcea2919

GitHub Gist Type: Secret

ghp\_8HR7Y9HjQTHRQg4wapP63tJOyz1blG2gBuDY

**Setting Sync: Sync: Загрузка Завершена. Gist ID: cd557bee87479c9dcbe5a7d8d0b768d0.**

**crkflbot**

**Sync: ist ID: af868ab1e3a53ab2a72fdc75270c5f44**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **branch**  **fetch**  **pull**  **push**  **remote** | **ветка**  **извлечь**  **тянуть**  **толкать**  **удаленный** |  |  |  |  |
|  |  |  |  |  |  |

# **Плагины**

Можно так же устанавливать сразу несколько пакетов, указывая их имена через запятую (чтобы не устанавливать по одному). Выполняется это через Preferences > Package Control > Package Control: Advanced Install Package.

**SCSS**

**AutoFileName**

**BracketHighlighter**

**ColorHighliter**

**ColorPiker**

**Emmet**

**Snippets**

**Goto-CSS-Declaration**

**Autoprefix CSS**

**CSScomb**

**All Autocomplete**

**Auto Complete Tag –** работа с тегами!!!

**FindKeyConflicts –** поискконфликта горячих клавиш

**Live Server –** позволяет автоматически перезагружать страницу

**LiveReload**

LiveReload —это утилита, которая позволяет автоматически перезагружать страницу в браузере при изменение её кода и ресурсов (html, css, js, images, etc) на сервере. Кроме того, LiveReload позволяет применять изменения в CSS и JavaScript без перезагрузки страницы.

Для работы вставить скрипт в теге <head>:

**<script>document.write('<script src="http://' + (location.host || 'localhost').split(':')[0] + ':35729/livereload.js?snipver=1"></' + 'script>')</script>**

**Tag** —Перенос классов в css

Для этого можно использовать специальный плагин Tag. Если что,это метод его установки в Sublime Text 3. 1.Скачиваем сам плагин : <http://fls.guru/files/Tag.zip> 2.Скачаный архив нужно распаковать в следующую папку: C:\users\Твоё имя пользователя\AppData\Roaming\Sublime Text 3\Packeges 3.И биндим клавиши для удобной работы: Preferences->Key bindings

[ { "keys": ["ctrl+`"], "command": "tag\_classes" } ]

**eCSStractor** —для копирования классов по БЭМ [https://fls.guru/files/ecsstractor.zip](https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqblh4TjJxNjhnazdmQ3VpcjlTaUhnbmx2cUZaQXxBQ3Jtc0tsRjJKN2VCc3RiLUdrQjNMRWVNZElaOHJSNndzX1FJdW1zYm9MdVJCTExtUXJrVlphcFBtdzk4NURSTl9IWFY4bHUxcDhMejFILTZPZXJTOVNKTW1oQTdJcWhCQ2g3Q205ZUNGblR5Z0RONUdmV1hRQQ&q=https%3A%2F%2Ffls.guru%2Ffiles%2Fecsstractor.zip&v=HihYQVuH64U)

<https://www.youtube.com/watch?v=HihYQVuH64U&list=PLM6XATa8CAG4F9nAIYNS5oAiPotxwLFIr&index=18>

**Settings Sync –** сохранение настроек **VScode** в облаке

**GitLens –** Связь проекта с GitHub

**Sass –** подсветка Sass

P.S.

Спасибо за видео, Женя! Была на Atom, перешла на Sublime :) Если что, для macOS последний плагин устанавливается следующим образом: нужно через Finder --> Переход (сверху) вызвать Библиотеки, зажав в открытом "Переход" клавишу Tab, и там по пути ~/Library/Application Support/Sublime Text 3/Packages установить папку Tag, которую Женя архивом прикрепил

# **Figma**

**font fascia –** плагин шрифтов

# **Теги**

Парные, блочные (занимает всю ширину страницы), не блочные

Блочные

<h1> - заголовок (только 1 раз на странице)

<p> - параграф (абзац)

<div>- блочный

<ol>- нумерованный ordered list

<ul>- маркированный unordered list

<li>- маркированный list item (содержимое списка)

(пример: ol>li\*10, Tab - создание списка с 10 элементами)

<form>- оболочка формы

Строчные

<strong> - выделение текста

<span> - строчный (не блочный) чтобы вёрстка не разваливалась

<a> - ссылка

<br> - отступ

<img> - позволяет добавить картинку в разметку

<input> - прием данных от пользователя

**.container>.header+.content>ul>li\*5{item$}**

**.container>.header>h1{Hello world}+.content>ul>li\*5{item$}**

**<тег атрибут1**="значение1" **атрибут2**="значение2"**>**

**<img src="keks.png" width="200" height="100">**

**<img src="img/like.png" alt="" />**

**<a target="\_blank" href="http://google.com">**

атрибут **target="\_blank"** – для открывания в новом окне

атрибут **alt="**Ой картинки нет (**"** – для комментария при отсутствии катринки

CSS (расшифровывается как Cascading Style Sheets — «каскадные таблицы стилей»)

Формы

**<form action=**"#" **method=**"get" (малое к-во данных) **method=**"post"(большое к-во данных) **enctype=**"multipart/form-data"**></form>**

**<input type="**text**" name="**username**" value="**+375**" />**

**<input type="**password**" name="**userpass**" value="" />**

**<input type="**tel**" >** - для телефона

**<body>**

**<form action="#" method="get" method="post" enctype="multipart/form-data"></form>**

**<input type="text" name="username" value="+375" />**

**<input placeholder="Подпись поля" type="password" name="userpass" value="" />**

**<input tabindex="1" placeholder="Подпись поля" type="password" name="userpass" value="" />**

**<input disabled placeholder="Подпись поля" type="password" name="userpass" value="" />**

**<div>**

**<input type="text" name="username" value="+375" />**

**</div>**

**<div>**

**<input type="password" name="userpass" value="" />**

**</div>**

**<div>**

**<p><input tabindex="2" checked type="radio" name="beer" value="lager" /> Светлое пиво</p>**

**<p><input tabindex="2" type="radio" name="beer" value="dark" /> Темное пиво</p>**

**</div>**

**<br />**

**<div>**

**<p><input type="radio" name="beer2" value="lager" /> Светлое пиво</p>**

**<p><input checked type="radio" name="beer2" value="dark" /> Темное пиво</p>**

**</div>**

**<br />**

**<div>**

**<p><input type="checkbox" name="car" value="yes" /> Наличие авто</p>**

**<p><input checked type="checkbox" name="car2" value="yes" /> Наличие авто 2</p>**

**<p><input type="checkbox" name="car3" value="yes" /> Наличие авто 3</p>**

**<p><input checked type="checkbox" name="car4" value="yes" /> Наличие авто 4</p>**

**</div>**

**<br />**

**<div>**

**<p><input type="file" name="foto" /></p>**

**</div>**

**<br />**

**<div>**

**<textarea name="text" id="" cols="30" rows="10"></textarea>**

**<textarea name="text"></textarea>**

**</div>**

**<br />**

**<div>**

**<select name="color">**

**<option value="red">Красный</option>**

**<option value="yellow">Желтый</option>**

**<option value="blue">Голубой</option>**

**</select>**

**</div>**

**<br />**

**<div>**

**<select multiple name="color">**

**<option value="red">Красный</option>**

**<option selected value="yellow">Желтый</option>**

**<option value="blue">Голубой</option>**

**</select>**

**</div>**

**<br />**

**<div>**

**<button type="submit"></button> <!--Отправится вся форма на сервер-->**

**</div>**

**<br />**

**<div>**

**<button type="reset"></button> <!--Очистится вся форма-->**

**</div>**

**</body>**

<p class="feature-furniture">…</p>

color: red; /\* красный цвет текста \*/

padding: 10px; /\* внутренний отступ размером 10 пикселей \*/

CSS-правило. Можно сказать, что правило — это группа свойств и их значений, которая целиком применяется к тем тегам, на которые указывает селектор.

селектор {

свойство: значение;

свойство: значение;

}

Селектор указывает, к каким тегам применятся свойства из CSS-правила. Проще всего работают селекторы по тегам: они выбирают все теги с подходящим именем. Например:

p { color: red; }

В примере селектором является p, и он выбирает все теги с именем p (то есть теги <p>), а теги с другим именем, например h1, не выбирает.

На нашем сайте подзаголовки размечены тегами с именем h2, и нам нужно оформить *все* эти теги. Селектор по тегу для этого отлично подойдёт, ведь он выберет все эти теги.

# **Селекторы**

Селектор – это обращение.

Селектор находится в начале CSS-правила, до фигурных скобок, и определяет к каким HTML-элементам применятся свойства и значения из правила. Вспомните пример:

.feature-kitten {

padding-top: 60px; /\* отступ сверху \*/

background-image: url("img/bottle.svg"); /\* фоновая картинка \*/

}

Строка .feature-kitten — это селектор. Она говорит браузеру применить список свойств (отступ и фон) ко всем элементам, у которых атрибут class имеет значение feature-kitten.

Простейшие (и самые популярные) селекторы — это селекторы по тегам и по классам. Селекторы по тегам содержат имя тега без символов < и > и применяются ко всем подходящим тегам. Селекторы по классам начинаются с точки, за которой идёт имя класса, и применяются ко всем тегам с подходящим атрибутом class. Например:

h1 { color: red; } /\* выберет все заголовки 1 уровня \*/

.info { color: blue; } /\* выберет только элементы с классом info \*/

Если у CSS-правил отличаются только селекторы, а свойства и значения одинаковые, то их можно сгруппировать через запятую. Например:

h1, .danger { color: red; }

/\* То же самое, что и \*/

Итак, наш план. Вы создаёте новое пустое CSS-правило для подзаголовков:

*нужное\_имя\_тега* {

}

Чем отличаются элементы обычного списка и списка преимуществ? Классом! У элементов списка преимуществ есть класс advantages-item. То есть нам достаточно выбрать теги с этим классом. И для этого есть специальный селектор, селектор по классу, например:

.*advantages-item* {…}

На самом деле у HTML-элемента может быть сколько угодно классов, в этом случае они перечисляются в атрибуте class через пробел, например:

<li class="product">Товар</li>

<li class="product hit">Товар, а ещё хит продаж</li>

<li class="product hit sale">Товар, хит продаж и со ски-и-идкой!</li>

Использование нескольких классов — это типовой приём, который ещё называется миксованием классов. Обычно его используют так: в один класс выносят общее оформление, а в дополнительных классах описывают его модификации. В примере выше размеры карточек товаров можно описать в CSS-правиле .product, а особый фон для хита продаж — в правиле .hit. В общем, удобный приём, помогает сократить дублирование кода.

Чтобы задать всем товарам один цвет фона, а хитам другой, можно сделать так:

.product {

… /\* общие размеры и отступы \*/

background-color: gray; /\* серый фон по умолчанию \*/

}

.hit {

background-color: yellow; /\* жёлтый фон у хитов \*/

}

Оба CSS-правила подействуют на второй товар, ведь у него есть два подходящих класса. Но чтобы цвет фона стал жёлтым, правило .hit должно находиться в коде ниже правила .product. В этом случае свойство background-color из второго правила *переопределит* значение свойства background-color из первого правила.

# **Синтаксис и обнуление**

**css\_1**

**css\_1\_2**

**css\_1\_3**

# **Свойства шрифта**

**css\_2**

**css\_2\_верстка**

font-family: "Имя шрифта", "Имя шрифта", тип шрифта

Устанавливает семейство шрифта

Типы шрифта:

serif — шрифты с засечками (антиквенные), типа Times;

sans-serif — рубленные шрифты (шрифты без засечек или гротески), типичный представитель — Arial;

cursive — курсивные шрифты;

fantasy — декоративные шрифты;

monospace — моноширинные шрифты, ширина каждого символа в таком семействе одинакова (шрифт Courier).

font-size

Определяет размер шрифта элемента.

font-style:

Определяет начертание шрифта — обычное, курсивное или наклонное.

normal - Обычное начертание текста.

italic - Курсивное начертание.

oblique - Наклонное начертание. Курсив и наклонный шрифт при всей их похожести не одно и то же. Курсив это специальный шрифт имитирующий рукописный, наклонный же образуется путем наклона обычных знаков вправо.

font-weight:

Устанавливает насыщенность шрифта.

100 - thin

300 - lite

400 - normal

500 - medium

600 - semibold

700 - bold

900 - black

# **Свойства текста**

color

Определяет цвет текста. Для задания цветов обычно используются числа в шестнадцатеричном коде,

либо с помощью RGB.

text-align

Определяет горизонтальное выравнивание текста в пределах элемента.

center - Выравнивание текста по центру.

justify - Выравнивание по ширине, что означает одновременное выравнивание по левому и правому краю. Чтобы произвести это действие браузер в этом случае добавляет пробелы между словами.

left - Выравнивание текста по левому краю.

right - Выравнивание текста по правому краю.

text-decoration

Добавляет оформление текста в виде его подчеркивания, перечеркивания, линии над текстом и мигания.

Одновременно можно применить более одного стиля, перечисляя значения через пробел. Это свойство наследуется и может принимать значения:

line-through - Создает перечеркнутый текст.

overline - Линия проходит над текстом.

underline - Устанавливает подчеркнутый текст.

none - Отменяет все эффекты, в том числе и подчеркивания у ссылок, которое задано по умолчанию.

text-shadow: горизонтальное\_смещение вертикальное\_смещение размер цвет;

Добавляет тень к тексту.

Прмер text-shadow: 1px 1px 1px #000;

text-transform

Управляет преобразованием текста в заглавные или прописные символы.

capitalize - Первый символ каждого слова в предложении будет заглавным. Остальные символы свой вид не меняют.

lowercase - Все символы текста становятся строчными (нижний регистр).

uppercase - Все символы текста становятся прописными (верхний регистр).

none - Не меняет регистр символов.

text-indent

Устанавливает величину отступа первой строки блока текста. При меняется, если нам нужно создать что-то типа красной строки.

letter-spacing

Определяет интервал между символами (буквами). Используется когда нужно разрядить текст.

line-height

Устанавливает межстрочный интервал текста. Широко применяется при верстке текста.

white-space

Управляет свойствами пробелов между словами. Пименяется в основном со значением

nowrap которое запрещает перенос строки. Таким образом весь текст отображается в одну строку и не ломается. Значение normal вернет все как было.

word-spacing

Устанавливает интервал между словами.

# **Свойства блоков - отступы, размеры, характеристики**

**css\_3**

**css\_3\_2**

box-sizing

Применяется для изменения алгоритма расчета ширины и высоты элемента.

Свойство наследуется.

content-box - Основывается на стандартах CSS, при этом свойства width и height задают ширину и высоту контента и не включают в себя значения отступов, полей и границ.

border-box - Свойства width и height включают в себя значения полей и границ, но не отступов (margin). Эта модель используется браузером Internet Exporer в режиме несовместимости.

padding-box - Свойства width и height включают в себя значения полей, но не отступов (margin) и границ (border).

=========================================

**padding**

Внутренний отступ элемента

При указании поля в процентах, значение считается от ширины родителя элемента.

Свойство не наследуется.

padding: со\_всех\_сторон;

padding: сверху справа снизу слева;

padding: сверху\_снизу справа\_слева;

padding: сверху справа\_слева снизу;

Отступ сверху и снизу не действует на срочные теги

=========================================

**margin**

Внешний отступ элемента

При указании поля в процентах, значение считается от ширины родителя элемента.

Свойство не наследуется.

Значение может быть как положительным, так и отрицательным числом.

margin: со\_всех\_сторон;

margin: сверху справа снизу слева;

margin: сверху\_снизу справа\_слева;

margin: сверху справа\_слева снизу;

Отступ сверху и снизу не действует на срочные теги

=========================================

width

Устанавливает ширину блочных тегов и некоторыйх строчных (например img)

Свойство не наследуется.

width:100px;

width:10%;

=========================================

max-width

Устанавливает максимальную ширину блочных тегов и некоторыйх строчных (например img)

=========================================

min-width

Устанавливает минимальную ширину блочных тегов и некоторыйх строчных (например img)

=========================================

height:

Устанавливает высоту блочных тегов и некоторыйх строчных (например img)

Свойство не наследуется.

height:100px;

height:10%;

=========================================

min-height

Свойство не наследуется.

=========================================

max-height

Свойство не наследуется.

=========================================

overflow

Управляет отображением содержания блочного элемента

visible - Отображается все содержание элемента, даже за пределами установленной высоты и ширины.

hidden - Отображается только область внутри элемента, остальное будет скрыто.

scroll - Всегда добавляются полосы прокрутки.

auto - Полосы прокрутки добавляются только при необходимости.

=========================================

display:

Многоцелевое свойство, которое определяет, как элемент должен быть показан в документе.

Свойство не наследуется.

block - Элемент показывается как блочный. Применение этого значения для встроенных элементов, например тега <span>, заставляет его вести подобно блокам — происходит перенос строк в начале и в конце содержимого.

inline - Элемент отображается как встроенный. Использование блочных тегов, таких как <div> и <p>, автоматически создает перенос и показывает содержимое этих тегов с новой строки. Значение inline отменяет эту особенность, поэтому содержимое блочных элементов начинается с того места, где окончился предыдущий элемент.

inline - block - это значение генерирует блочный элемент, который обтекается другими элементами веб-страницы подобно встроенному элементу. Фактически такой элемент по своему действию похож на встраиваемые элементы (вроде тега <img>). При этом его внутренняя часть форматируется как блочный элемент, а сам элемент — как встроенный.

none - Временно удаляет элемент из документа. Занимаемое им место не резервируется и веб-страница формируется так, словно элемента и не было.

=========================================

# **Свойства блоков - стили оформления**

**css\_4**

=========================================

border

Универсальное свойство border позволяет одновременно установить толщину,

стиль и цвет границы вокруг элемента.

border: 1px solid #000; (размер стиль цвет)

Основные стили -solid dotted dashed

=========================================

border-radius

Устанавливает радиус скругления уголков блока.

border-radius:50%; - кргуг

Можно использовать вместе с overflow: hidden;

=========================================

outline

Универсальное свойство, одновременно устанавливающее цвет,

стиль и толщину внешней границы на всех четырех сторонах элемента.

В отличие от линии, задаваемой через border, свойство outline

не влияет на положение блока и его ширину.

Также нельзя задать параметры линии на отдельных сторонах элемента,

outline применяется сразу ко всем четырём сторонам.

=========================================

box-shadow

Добавляет тень к элементу.

box-shadow: сдвиг\_по\_гориз сдвиг\_по\_верт радиус расстояние

Можно добавить много теней. На тень влияет свойство border-radius.

=========================================

opacity

Определяет уровень прозрачности элемента.

Отличие opacity:0; от display:none; в том что блок не убирается из верстки а только становиться прозрачным, то есть занимаемое им место остается. Так же, с прозрачными элементами все еще можно взаимодействовать, например кликать по ссылкам.

=========================================

visibility

Предназначен для отображения или скрытия элемента.

Отличие visibility: hidden; от display:none; в том что блок не убирается из верстки а только скрывается, то есть занимаемое им место остается.

Отличие visibility: hidden; от opacity:0; в том что блок скрывается и взаимодействовать с ним нельзя.

=========================================

# **Свойства блоков - свойство background**

**css\_5**

**css\_5\_\_**

background:

Универсальное свойство background позволяет установить одновременно несколько характеристик фона а именно:

=======================================================

background-color - Определяет цвет фона элемента.

=======================================================

background-image - Устанавливает фоновое изображение для элемента.

Так же значением свойства можно указать градиент:

линейный:

background:linear-gradient(to top, #fefcea, #f1da36);

радиальный:

background:radial-gradient(ellipse at center, rgba(30,87,153,1) 0%,rgba(41,137,216,1) 50%,rgba(32,124,202,1) 51%,rgba(125,185,232,1) 100%);

Подробнее о градиентах:

http://htmlbook.ru/css3-na-primerakh/lineinyi-gradient

Инструмент создания готового кода:

https://www.colorzilla.com/gradient-editor/

=======================================================

background-position - Задает начальное положение фонового изображения, установленного с помощью свойства background-image.

=======================================================

background-repeat - Определяет, как будет повторяться фоновое изображение, установленное с помощью свойства background-image.

no-repeat - Устанавливает одно фоновое изображение в элементе без его повторений, положение которого определяется свойством background-position (по умолчанию в левом верхнем углу). Аналогично no-repeat no-repeat.

repeat - Фоновое изображение повторяется по горизонтали и вертикали. Аналогично repeat repeat.

repeat-x - Фоновый рисунок повторяется только по горизонтали. Аналогично repeat no-repeat.

repeat-y - Фоновый рисунок повторяется только по вертикали. Аналогично no-repeat repeat.

inherit -Наследует значение родителя.

space - Изображение повторяется столько раз, чтобы полностью заполнить область; если это не удаётся, между картинками добавляется пустое пространство.

round - Изображение повторяется так, чтобы в области поместилось целое число рисунков; если это не удаётся сделать, то фоновые рисунки масштабируются.

=======================================================

background-attachment - Устанавливает, будет ли прокручиваться фоновое изображение вместе с содержимым элемента.

fixed - Делает фоновое изображение элемента неподвижным.

scroll - Позволяет перемещаться фону вместе с содержимым.

inherit - Наследует значение родителя.

local - Фон фиксируется с учётом поведения элемента. Если элемент имеет прокрутку, то фон будет прокручиваться вместе с содержимым, но фон выходящий за рамки элемента остаётся на месте.

=======================================================

background-size - Масштабирует фоновое изображение согласно заданным размерам.

<значение> - Задает размер в любых доступных для CSS единицах.

<проценты> - Задает размер фоновой картинки в процентах от ширины или высоты элемента.

auto - Если задано одновременно для ширины и высоты (auto auto), размеры фона остаются исходными; если только для одной стороны картинки (100px auto), то размер вычисляется автоматически исходя из пропорций картинки.

cover - Масштабирует изображение с сохранением пропорций так, чтобы его ширина или высота равнялась ширине или высоте блока.

contain - Масштабирует изображение с сохранением пропорций таким образом, чтобы картинка целиком поместилась внутрь блока.

Если установлено одно значение, оно задает ширину фона, второе значение принимается за auto. Пропорции картинки при этом сохраняются. Использование двух значений через пробел задает ширину и высоту фоновой картинки.

=======================================================

Свойство background позволяет задать несколько изображений через запятую,

причем со своими настройками позиционарования маштабирования и прокрутки

=======================================================

# **CSS псевдоклассы. Псевдокласс HOVER и другие**

**css\_6**

Синтаксис, то есть правило записи псевдоклассов прост, мы пишем селектор класса либо селектор типа

ставим двоеточие, пишем тот или иной псевдосласс и уже после этого открываем фигурные скобки и пишем нужные CSS параметры:

Селектор: Псевдокласс { параметры стиля }

================================================================================================================

Отлично, теперь рассмотрим псевдоклассы состояния.

:hover

Срабатывает при наведении на элемент, часто применяется применяется как для ссылок так и для любого другого элемента.

:active

Срабатывает при нажатии на элемент. В основном применяется к ссылкам и кнопкам.

:visited

Срабатывает для посещенных ссылок

:focus

Срабатывает при получении элементом фокуса. Часто применяется к елементам форм. Например инпутам.

================================================================================================================

Теперь рассмотрим псевдоклассы положения в коде.

:first-child

Обращение к первому элементу в блоке

:last-child

Обращение к последнему элементу в блоке

:nth-child(порядковый номер элемента)

Обращение к конкретным элементам в блоке

odd - Обращается к элементам с нечётными номерами

even - Обращается к элементам с чётными номерами

================================================================================================================

# **CSS псевдоэлементы. Псевдоэлементы BEFORE и AFTER**

**css\_7**

Псевдоэлементы – это селекторы, которые определяют область элементов, которая изначально

отсутствует в дереве документа. Эта область создается искусственно с помощью CSS.

Псевдоэлемент :first-line задает стиль первой строки форматированного текста.

Длина этой строки зависит от многих факторов, таких как используемый шрифт,

размер окна браузера, ширина блока, языка и т.д. В правилах стиля допустимо

использовать только свойства, относящиеся к шрифту, изменению цвета текста и фона.

=================================================================================

Псевдоэлемент :first-letter определяет стиль первого символа в тексте элемента,

к которому добавляется.

К этому псевдоэлементу могут применяться только стилевые свойства,

связанные со свойствами шрифта, полями, отступами, границами, цветом и фоном.

=================================================================================

Псевдоэлемент :before применяется для отображения желаемого контента до содержимого элемента,

к которому он добавляется. Работает совместно со свойством content.

При добавлении :before к блочному элементу, значение свойства display может быть только: block, inline, none, list-item.

Все остальные значения будут трактоваться как block.

При добавлении :before к встроенному элементу, display ограничен значениями inline и none.

Все остальные будут восприниматься как inline.

=================================================================================

Псевдоэлемент :after, который используется для вывода желаемого текста после содержимого элемента, к которому он добавляется. Псевдоэлемент :after работает совместно со свойством content.

При добавлении :after к блочному элементу, значение свойства display может быть только:

block, inline, none, list-item. Все остальные значения будут трактоваться как block.

При добавлении :after к встроенному элементу, display ограничен значениями inline и none.

Все остальные будут восприниматься как inline.

=================================================================================

::-ms-clear {}

Задаёт стиль кнопки для очистки текстового поля. Исходно эта кнопка не видна,

она появляется в правой части поля только при вводе текста.

Только IE

=================================================================================

::-moz-focus-inner{}

Задаёт стиль внутренней части элемента

Только FireFox

=================================================================================

# **CSS позиционирование (CSS position)**

**css\_8**

Свойство position

Устанавливает способ позиционирования элемента относительно окна браузера или других объектов на веб-странице.

Вспомогательные свойства left, right, top и bottom управляют положением элемента,

а z-index управляет наложением друг на друга по оси Z.

Значения:

static - элемент позиционируется относительно родителя и соседних элементов

relative - элемент позиционируется как static но можно двигать его относительно своего положения

absolute - элемент позиционируется относительно ближайшего родителя с relative, absolute, fixed и sticky

fixed - элемент позиционируется относительно окна браузера

sticky - элемент позиционируется как static но когда верхняя граница элемента будет находиться на расстоянии, указанном в параметре top, от верхней границы окна браузера (либо указанном в параметре bottom от нижней),

он останется в этом положении относительно окна до тех пор, пока его нижняя граница не упрется в другой sticky элемент или конец родительского элемента

inherit - берет значение от родителя

==========================================================

Свойства:

top, bottom, left и right - управляют положением элемента сверху снизу слева и справа.

Значения могут указываются в том числе в процентах и могут иметь отрицательные значения.

Работают с элементами, к которым применен один из параметров

position: relative, position: absolute, position: fixed и position: sticky (только top или bottom)

==========================================================

Свойство: z-index

Управляет наложением друг на друга элементов с параметрами

position: relative, position: absolute, position: fixed по оси Z

# **Препроцессор SASS/SCSS за 38 минут. Что такое SASS, SCSS**

**hellosass**

Сайт препроцессора SASS

<https://sass-scss.ru/guide/>

Содержание:

[00:00](https://www.youtube.com/watch?v=Mrq2ora_p0o&list=PLM6XATa8CAG4F9nAIYNS5oAiPotxwLFIr&index=15&t=0s) – Вступление

[02:03](https://www.youtube.com/watch?v=Mrq2ora_p0o&list=PLM6XATa8CAG4F9nAIYNS5oAiPotxwLFIr&index=15&t=123s) – Синтаксис

[04:45](https://www.youtube.com/watch?v=Mrq2ora_p0o&list=PLM6XATa8CAG4F9nAIYNS5oAiPotxwLFIr&index=15&t=285s) – Вложенность

[16:35](https://www.youtube.com/watch?v=Mrq2ora_p0o&list=PLM6XATa8CAG4F9nAIYNS5oAiPotxwLFIr&index=15&t=995s) – Переменные

[18:36](https://www.youtube.com/watch?v=Mrq2ora_p0o&list=PLM6XATa8CAG4F9nAIYNS5oAiPotxwLFIr&index=15&t=1116s) – Импорт

[22:41](https://www.youtube.com/watch?v=Mrq2ora_p0o&list=PLM6XATa8CAG4F9nAIYNS5oAiPotxwLFIr&index=15&t=1361s) – Шаблоны

[29:44](https://www.youtube.com/watch?v=Mrq2ora_p0o&list=PLM6XATa8CAG4F9nAIYNS5oAiPotxwLFIr&index=15&t=1784s) – Миксины

[32:41](https://www.youtube.com/watch?v=Mrq2ora_p0o&list=PLM6XATa8CAG4F9nAIYNS5oAiPotxwLFIr&index=15&t=1961s) – Математика

[34:59](https://www.youtube.com/watch?v=Mrq2ora_p0o&list=PLM6XATa8CAG4F9nAIYNS5oAiPotxwLFIr&index=15&t=2099s) - Комментирование

=================================================

ОЧЕНЬ ценю твой ЛАЙК и КОММЕНТ!

Поддержать канал: https://www.patreon.com/freelancerlifestyle

Telegram канал: https://t.me/freelancer\_lifestyle или https://teleg.run/freelancer\_lifestyle

Instagram: https://www.instagram.com/freelancer.lifestyle/

Facebook: https://www.facebook.com/freelancerlifestyle/

Telegram ЧАТ: https://t.me/flschat или https://teleg.run/flschat

# **Подключение шрифтов. Google Fonts. CSS font-face**

**addfonts**

**О подключении шрифтов через Google Fonts:** [1:55](https://www.youtube.com/watch?v=GwA0BN5RgB0&t=115s)

**1-й вариант подключения шрифта (для HTML, CSS):** [4:08](https://www.youtube.com/watch?v=GwA0BN5RgB0&t=248s)

**2-й вариант подключения шрифта (@import для SASS/SCSS):** [4:46](https://www.youtube.com/watch?v=GwA0BN5RgB0&t=286s)

**Использование подключенного шрифта:** [5:28](https://www.youtube.com/watch?v=GwA0BN5RgB0&t=328s)

**Скачивание шрифта с nomail.com.ua:** [9:45](https://www.youtube.com/watch?v=GwA0BN5RgB0&t=585s)

**Конвертация скачанного шрифта:** [11:20](https://www.youtube.com/watch?v=GwA0BN5RgB0&t=680s)

**3-й вариант подключения шрифта (@font-face):** [13:21](https://www.youtube.com/watch?v=GwA0BN5RgB0&t=801s)

**Путь к скачанному шрифту:** [16:34](https://www.youtube.com/watch?v=GwA0BN5RgB0&t=994s)

**Использование подключенного шрифта:** [17:42](https://www.youtube.com/watch?v=GwA0BN5RgB0&t=1062s)

**Женя, ценю то, что ты делаешь. Поэтому вот ТАЙМКОДЫ :)**

[**00:00**](https://www.youtube.com/watch?v=GwA0BN5RgB0&t=0s) **– Вступление. Зачем подключать шрифты?**

[**00:37**](https://www.youtube.com/watch?v=GwA0BN5RgB0&t=37s) **– Подготовка к уроку**

[**01:56**](https://www.youtube.com/watch?v=GwA0BN5RgB0&t=116s) **– Способ 1. Подключение с помощью Google Fonts**

[**09:04**](https://www.youtube.com/watch?v=GwA0BN5RgB0&t=544s) **– Способ 2. Когда Google Fonts бессилен**

[**19:22**](https://www.youtube.com/watch?v=GwA0BN5RgB0&t=1162s) **– Домашнее задание. Заключение**

ШПАРГАЛКА

========================================================================

Каталог шрифтов:

https://nomail.com.ua/

Google Fonts:

https://fonts.google.com/

Конвертер шрифтов:

http://www.font2web.com/

========================================================================

@font-face {

font-family: 'Имя шрифта';

font-display: swap;

src: url("../fonts/файл шрифта.eot");

src: local("O"), url("../fonts/файл шрифта.woff") format("woff"),

url("../fonts/файл шрифта.ttf") format("truetype"),

url("../fonts/файл шрифта.svg") format("svg");

font-weight: normal;

font-style: normal;

}

========================================================================

font-family: "Имя шрифта", "Имя шрифта", тип шрифта

Устанавливает семейство шрифта

Типы шрифта:

serif — шрифты с засечками (антиквенные), типа Times;

sans-serif — рубленные шрифты (шрифты без засечек или гротески), типичный представитель — Arial;

cursive — курсивные шрифты;

fantasy — декоративные шрифты;

monospace — моноширинные шрифты, ширина каждого символа в таком семействе одинакова (шрифт Courier).

========================================================================

font-weight:

Устанавливает насыщенность шрифта.

100 - thin

200 - UltraLite

300 - lite

400 - normal

500 - medium

600 - semibold

700 - bold

800 - Heavy

900 - black

========================================================================

# **Методология БЭМ (Блок. Элемент. Модификатор)**

**bem**

Сайт методологии БЭМ

https://ru.bem.info/methodology/quick-start/

=================================================

Telegram канал: https://t.me/freelancer\_lifestyle или https://teleg.run/freelancer\_lifestyle

Instagram: https://www.instagram.com/freelancer.lifestyle/

Facebook: https://www.facebook.com/freelancerlifestyle/

Telegram ЧАТ: https://t.me/flschat или https://teleg.run/flschat

=================================================

**Блок** – это функционально независимый компонент страницы, который может быть повторно использован. Когда мы задаем блоку имя класса то должны отвечать на вопрос «что это?» а не «какой, как выглядит?».

=================================================

**Элемент** – это составная часть блока, которая не может использоваться в отрыве от него. Так же как для блока имя класса должно отвечать на вопрос «Что это?», но синтаксис записи будет следующим:

имякласса\_\_имяэлемента

=================================================

**Модификатор**.

Применяется для определения или уточнения внешнего вида, состояния или поведения блока либо элемента. Когда нужно выделить некий объект из множества таких же.

Имя класса модификатора должно отвечать на вопрос «какой?» «как выглядит?» «как себя ведет\состояние?»

Модификатор дописывается к классу блока либо элемента путем дублирования основного класса с добавлением одного нижнего подчеркивания и имени модификатора.

===================Вложенность блоков==============================

<!-- Блок `about` -->

<div class="about">

<!-- Вложенный блок `title` -->

<div class="title"></div>

<!-- Вложенный блок `subtitle` -->

<div class="subtitle"></div>

</div>

===================Вложенность элементов==============================

<!-- Верно -->

<!-- Блок `row` -->

<div class="row">

<!-- Элемент `column` -->

<div class="row\_\_column ">

<!-- Элемент `item` -->

<div class="row\_\_item">

</div>

</div>

</div>

<!-- Неверно -->

<!-- Блок `row` -->

<div class="row">

<!-- Элемент `column` -->

<div class="row\_\_column ">

<!-- Элемент `item` -->

<div class="row\_\_column\_\_item">

</div>

</div>

</div>

====================Принадлежность элементов===================================

<!-- Верный пример -->

<!-- Блок `about` -->

<div class=" about">

<!-- Элемент `title` -->

<div class="about\_\_title"></div>

<!-- Элемент `subtitle` -->

<div class=" about\_\_subtitle"></div>

</div>

<!-- Неверный пример -->

<!-- Блок `about` -->

<div class=" about">

<!-- Элемент `title` -->

<div class="about\_\_title"></div>

</div>

<!-- Элемент `subtitle` -->

<div class=" about\_\_subtitle"></div>

====================Необязательность элементов===================================

<!-- Блок `about` -->

<div class=" about">

<!-- Вложенный блок `title` -->

<div class="title"></div>

<!-- Вложенный блок `subtitle` -->

<div class="subtitle"></div>

</div>

====================Модификатор===================================

<nav class="menu">

<a href="" class="menu\_\_item menu\_\_item\_active">HOME</a>

<a href="" class="menu\_\_item">ABOUT US</a>

<a href="" class="menu\_\_item">PORTFOLIO</a>

<a href="" class="menu\_\_item">CONTACT</a>

</nav>

====================Микс===================================

<!-- Блок `about` -->

<div class=" about">

<!-- Элемент `title` -->

<div class="about\_\_title title"></div>

<!-- Элемент `subtitle` -->

<div class=" about\_\_subtitle subtitle"></div>

</div>

# **HTML мета теги для SEO, адаптива, социальных сетей и для iPhone**

**meta**

МЕТА теги

Основные теги влияющие на отображение страницы

<!-- Кодировка страницы -->

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

С появлением HTML5 можно использовать

<meta charset="UTF-8">

<!-- Настройки адаптивности страницы -->

Фиксированная ширина

<meta name="viewport" content="width=1170">

Адаптивность

<meta name="viewport" content="width=device-width">

<!-- Отключает принудительную ссылку у номера телефона на iOS -->

<meta name="format-detection" content="telephone=no">

=================================================

SEO

<!-- Краткое описание страницы -->

<meta name="description" content=" ">

Пишем не более 140 символов.

<!-- Ключевые слова страницы -->

<meta name="keywords" content=" ">

Пишем не более 20ти слов. Ключевые фразы разделяем запятой.

<!-- Управление доступом поисковых роботов к странице -->

<meta name="robots" content="">

По умолчанию страница индексируется поисковыми система, то есть этот мета тег дополнительно указывать не нужно. Но если задача стоит запретить либо ограничить индексацию, то существуют следующие значения:

none – запретить индексацию страницы полностью.

noindex – запретить индексацию содержимого страницы.

nofollow – игнорировать ссылки в пределах веб-страницы.

noimageindex - запретить индексацию присутствующих на странице изображений

noarchive - запретить вывод в результатах поиска ссылки «Сохраненная копия»

nosnippet - запретить вывод в поисковой выдаче под названием страницы фрагмента текста, описывающего её содержание.

Значения указываются через запятую. Например:

Запрет индексации страницы полностью будет выглядеть так:

<meta name="robots" content="noindex, nofollow">

или так

<meta name="robots" content="none">

Запрет индексации ссылок и изображений на странице будет выглядеть так:

<meta name="robots" content="noimageindex, nofollow">

при этом текст страницы будет доступен к индексации.

=================================================

Технические

<!-- Автор страницы -->

<meta name="Author" content="Пупкин Василий Петрович">

<!-- Авторские права -->

<meta name="Copyright" content="Зимина Татьяна Юрьевна">

<!-- Адрес автора -->

<meta name="Address" content="Луна, кратер №97">

<!-- Редирект (перезагрузка) страницы. Задержка в секундах; url=Адрес сайта/страницы -->

<meta http-equiv="refresh" content="S; url=URL">

S= Задержка в секундах

URL= Адрес сайта/страницы

=================================================

Для социальных сетей

Facebook

Что бы настроить вид нашей страницы в посте мы пишем следующие мета теги:

<!-- локализация сайта, для русскоязычного сайта ru\_RU -->

<meta property="og:locale" content="ru\_RU">

<!-- тип контента, по умолчанию используется article -->

<meta property="og:type" content="article">

<!-- заголовок страницы, который будет выводится в записи социальной сети -->

<meta property="og:title" content="META теги">

<!-- описание страницы -->

<meta property="og:description" content="Описание страницы про META теги">

<!-- ссылка на изображение, которое будет публиковаться в записи -->

<meta property="og:image" content="http://fls.guru/meta/img/bg.jpg">

<!-- ссылка на текущую страницу -->

<meta property="og:url" content="http://fls.guru/meta/ ">

<!-- название сайта -->

<meta property="og:site\_name" content="Фрилансер по жизни">

Проверить разметку можно в валидаторе от Facebook https://developers.facebook.com/tools/debug/sharing/

Подробнее о протоколе Open Graph можно почитать в официальной документации по ссылке в описании. https://ruogp.me/

Для создания Twitter Cards мета теги будут иметь другой вид:

<!-- Тип карты, по умолчанию используется summary -->

<meta name="twitter:card" content="summary">

<!-- Имя/логин автора -->

<meta name="twitter:site" content="Жека">

<!-- Название страницы -->

<meta name="twitter:title" content="META теги">

<!-- Описание страницы -->

<meta name="twitter:description" content="про META теги">

<!-- Cсылка на изображение -->

<meta name="twitter:image" content="http://fls.guru/meta/img/bg.jpg">

Проверяем результат в валидаторе твиттер

https://cards-dev.twitter.com/validator

Подробнее о создании Twitter Cards можно почитать в официальной документации по ссылке в описании.

https://developer.twitter.com/en/docs/tweets/optimize-with-cards/guides/getting-started

# Испытание: поправил классы — спас люстру

В испытаниях вы проверяете, насколько хорошо усвоили материал. Задача — с помощью HTML и CSS получить в мини-браузере изображение, максимально похожее на образец.

В этом испытании вам нужно починить сломанную вёрстку интернет-магазина, используя знания из [части «Основы HTML и CSS»](https://htmlacademy.ru/courses/297): добавить правильные классы в разметку и поправить стили. Чтобы пройти испытание выполняйте по порядку следующие шаги:

В index.html:

1. Тегу <div> на 15 строке [добавьте класс](https://htmlacademy.ru/courses/297/run/5) intro-container.

Нажмите кнопку «Сравнить» и загляните во вкладки «Результат» и «Различия». Во вкладке «Результат» отображается то, как ваш код видит наш сервер. Во вкладке «Различия» — разница между образцом и результатом.

1. Третьей карточке с товаром (36 строка) [добавьте второй класс](https://htmlacademy.ru/courses/297/run/10) product-card-new. Обратите внимание, классы разделяются пробелом.

Снова нажмите «Сравнить», посмотрите на изменения во вкладках, а также уровень соответствия образца и результата (проценты слева от кнопки «Сравнить»).

В style.css:

1. В правило для класса intro-container (31 строка) [добавьте свойство](https://htmlacademy.ru/courses/297/run/7) background-image со значением url("img/index-background.jpg"). Не забудьте про точку с запятой!

Снова нажмите «Сравнить». Теперь у вас должно быть 92%. Если результат и образец совпадают больше, чем на 90%, испытание считается пройденным.

1. Чтобы достичь 100% прохождения испытания, подберите правильный размер шрифта для класса title (41 строка). Он отличается от того, что указан сейчас, всего на 1px!

Не забывайте нажимать кнопку сравнения после каждого изменения.

# Находим элемент с помощью querySelector

Итак, чтобы найти на странице элемент с классом page, мы должны написать:

document.querySelector('.page');

Чтобы вывести сообщение в консоль, нужно использовать console.log:

console.log('Привет от JavaScript!');

В результате этой инструкции в консоли появится сообщение Привет от JavaScript!

В консоль можно выводить не только текст, но и результаты выполнения инструкций. Например, найденный с помощью querySelector элемент:

console.log(document.querySelector('.page'));

# Убираем класс с помощью classList.remove

Мы убедились, что JavaScript действительно нашёл элемент с классом page. Пора заняться переключением тем. У найденного элемента есть ещё второй класс — light-theme, благодаря ему на странице включена светлая тема. Чтобы не было смешения стилей, когда мы переключим тему, этот класс у элемента нужно убрать. Для этого нам понадобится метод classList.remove. Используют его так:

элемент.classList.remove('класс');

Метод убирает с элемента тот класс, который указан в скобках. Обратите внимание, что мы не ставим точку перед именем класса в classList.remove. Это не селектор, JavaScript и так знает, что мы имеем дело с классом.

Обратите внимание, что буква «L» в названии метода заглавная. JavaScript чувствителен к регистру. Если написать classlist.remove, метод не сработает.

Чтобы выключить светлую тему, убрав класс light-theme у элемента page, используем инструкцию:

document.querySelector('.page').classList.remove('light-theme');

# Добавляем класс элементу с помощью classList.add

# Объявляем переменную

Мы переключили тему на странице, но строки с кодом получились длинными и трудночитаемыми. А ещё пришлось дважды искать с помощью querySelector один и тот же элемент. Мы его уже один раз нашли, зачем искать снова?

Выход есть! И имя ему — переменная.

Переменная — это способ сохранить данные, дав им понятное название.

В JavaScript переменную можно создать, или, как говорят программисты, **объявить**, с помощью ключевого слова let. За ним следует имя переменной. После объявления в переменную нужно записать, или **присвоить**, какое-то значение:

let variableName = 'Я значение переменной!';

Имя переменной может быть почти любым, но не должно начинаться с цифры, а из спецсимволов разрешены только '\_' и '$'. Кроме того, в JavaScript есть [зарезервированные слова](https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Lexical_grammar#_blank), которые нельзя использовать для именования переменных. Имена переменных чувствительны к регистру: header, Header и HEADER — это разные переменные. Но самое главное — чтобы переменная действительно делала код понятнее, её имя должно описывать то, что в ней хранится.

Сохранять в переменные можно что угодно, в том числе элементы:

let header = document.querySelector('header');

Когда в коде встречается переменная, браузер вместо её имени подставляет присвоенное ей значение. Благодаря этому мы можем написать, например, так:

console.log(header);

header.classList.add('new-class');

Обратите внимание, что let мы пишем только при объявлении переменной. После этого мы используем переменную, просто указывая её имя.

**Swift**

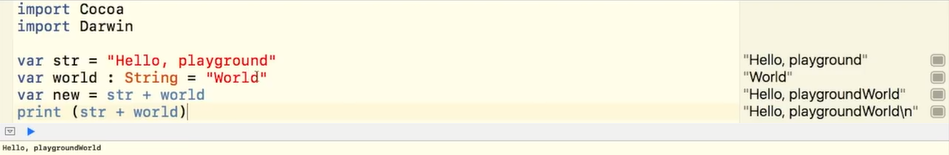
**Операторы**

Операторы делятся на унарные, бинарные и тернарные.

Язык Swift поддерживает все стандартные *операторы сравнения* из C:

* равно (a == b)
* не равно (a != b)
* больше (a > b)
* меньше (a < b)
* больше или равно (a >= b)
* меньше или равно (a <= b)

**Переменные 4:05**



**Переменные и их типы5:12**

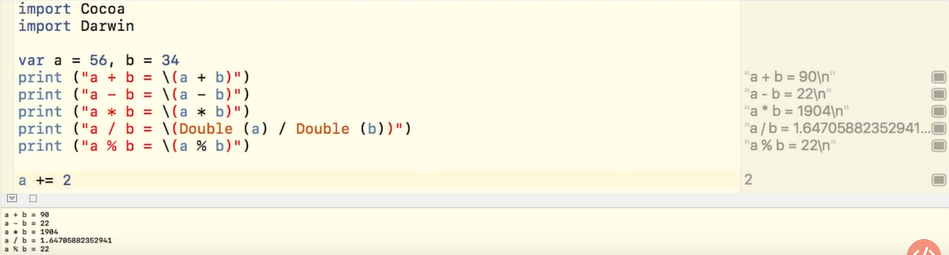
****

Можно через оператор **tipealias** создать свой тип:

typealias myNumber = Int

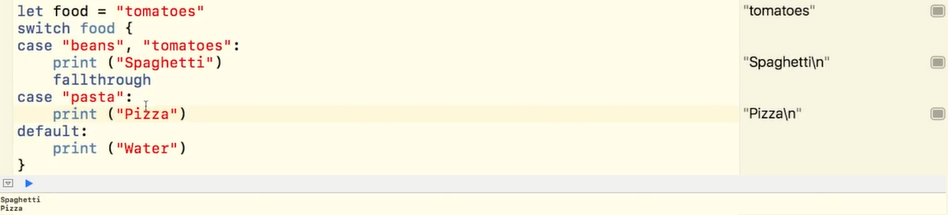
var chislo: myNumber = 10

**Математические операции 9:10**

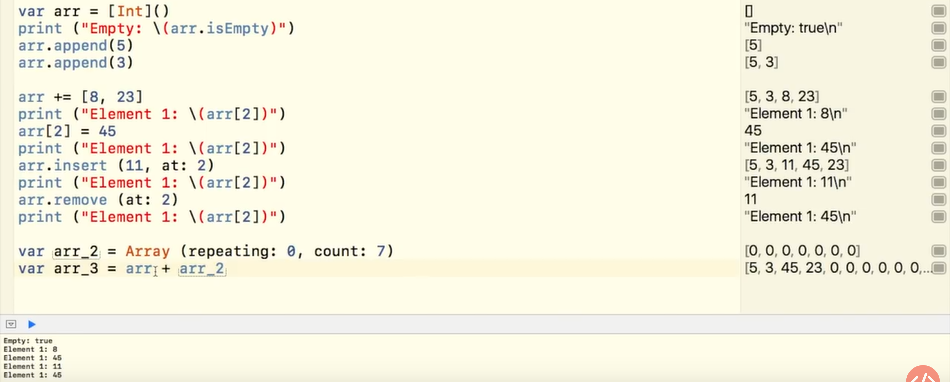
****

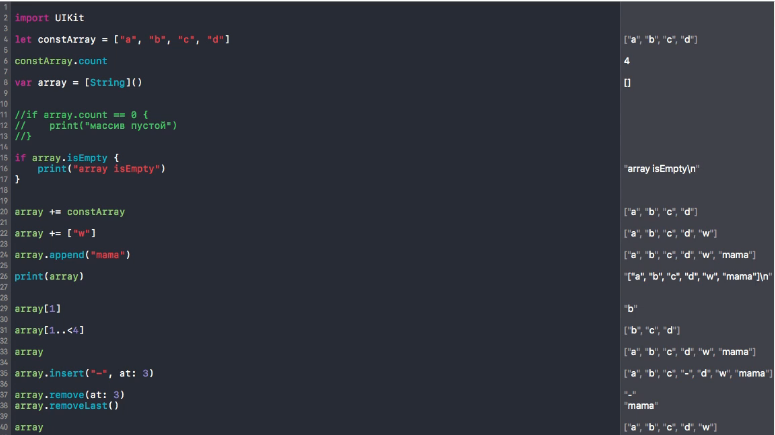
**Условные операторы 11:58**

****

****

**Массивы 20:14**





var fourDoubles = [Double](repeating: 0.0, count: 4)

print(fourDoubles)

var shoppingList = ["Bread", "Milk"]

shoppingList.append("Flour")

print(shoppingList)

var shoppingList = ["Bread", "Milk"]

shoppingList[0] = "Two apples"

print(shoppingList)

var shoppingList = ["Syrup", "Bread", "Milk", "Apples", "Soup"]

print("Before: \(shoppingList)")

shoppingList[1...2] = ["Bananas", "Oranges"]

print("After: \(shoppingList)")

let r = shoppingList.remove(at:0)

print("After: \(shoppingList)")

print(r)

var shoppingList = ["Syrup", "Bread", "Milk", "Apples", "Soup"]

let r = shoppingList.removeLast()

shoppingList.insert("New item",at:0)

var shoppingList = ["Bread", "Milk", "Syrup", "Apples"]

for item in shoppingList {

print(item)

}

В качестве альтернативы используйте метод enumerated () для итерации по массиву, когда вам нужен целочисленный индекс для каждого элемента в дополнение к его значению. Это возвращает кортеж для каждого элемента в массиве, который указывает индекс и значение этого элемента. Вы можете разложить кортеж на временные константы или переменные как часть итерации:

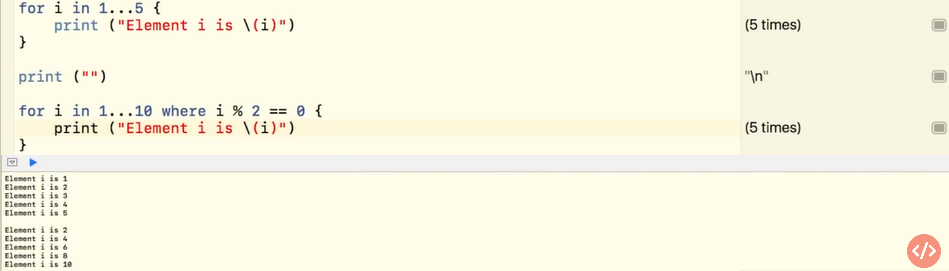
var shoppingList = ["Bread", "Milk", "Syrup", "Apples"]

for (index, value) in shoppingList.enumerated() {

print("Item \(index + 1): \(value)")

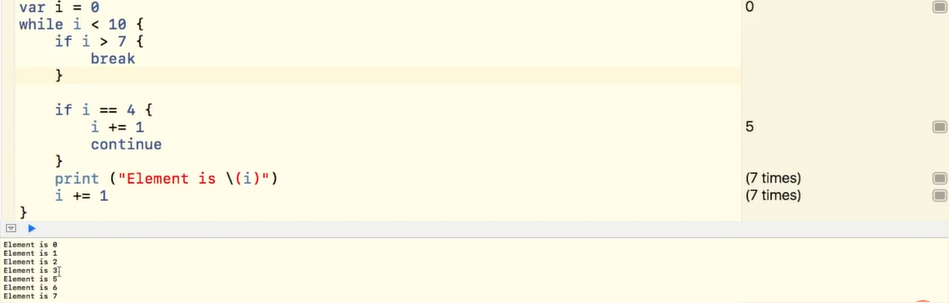
}

**Циклы 24:46**

****

Циклы в массивах **25:29**

****

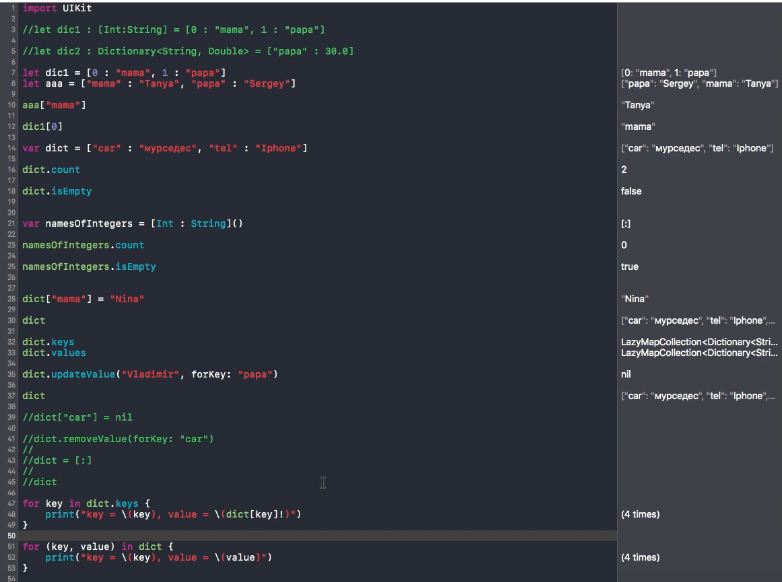


Цикл, который выполняется всегда хотя бы 1 раз **26:52**



**Словари (используются ключи) 27:38**

****



var airports = ["TOR": "Toronto", "NY": "New York"]

if let removedValue = airports.removeValue(forKey: "NY") {

print("The removed airport's name is \(removedValue).")

} else {

print("The airports dictionary does not contain a value for NY.")

}

var airports = ["TOR": "Toronto", "NY": "New York"]

for (airportCode, airportName) in airports {

print("\(airportCode): \(airportName)")

}

var airports = ["TOR": "Toronto", "NY": "New York"]

for airportCode in airports.keys {

print("Airport code: \(airportCode)")

}

for airportName in airports.values {

print("Airport name: \(airportName)")

}

**Кортеж (подобие массива, который нельзя менять) 30:30**

Кортежи группируют несколько значений в одно составное значение. Значения внутри кортежа могут быть любого типа, то есть, нет необходимости, чтобы они были одного и того же типа.

**Заметка**

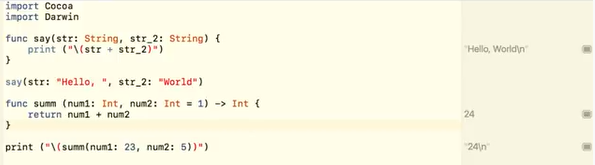
Кортежи полезны для временной группировки связанных значений. Они не подходят для создания сложных структур данных. Если ваша структура данных, вероятно, будет выходить за рамки временной структуры, то такие вещи лучше проектируйте с помощью классов или структур, вместо кортежей.



**Необязательные значения 31:54**

****

**Функция 33:28**



**Функция 33:28**

var result = 0

for i in 0...4 {

if i == 3 {

result += 10

} else {

result += i

}

print(result)

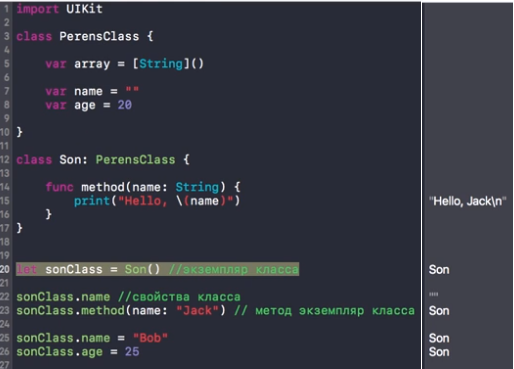
}

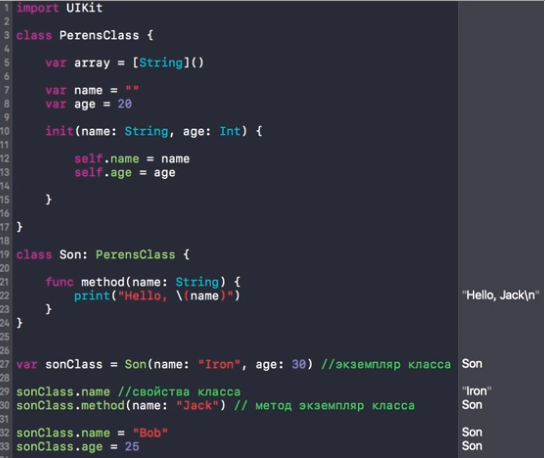
17

**Классы и Структуры**

Классы имеют дополнительную возможность, которых нет у структур:

* Наследование позволяет одному классу наследовать характеристики другого
* Приведение типов позволяет проверить и интерпретировать тип экземпляра класса в процессе выполнения
* Деинициализаторы позволяют экземпляру класса освободить любые ресурсы, которые он использовал
* Подсчет ссылок допускает более чем одну ссылку на экземпляр класса. Для получения дополнительной информации смотрите [Наследование](https://swiftbook.ru/doc/inheritance), [Приведение типов](https://swiftbook.ru/doc/type-casting), [Деинициализаторы и](https://swiftbook.ru/doc/deinitialization)[Автоматический подсчет ссылок](https://swiftbook.ru/doc/automatic-reference-counting).

Дополнительные возможности поддержки классов связаны с увеличением сложности. Лучше использовать структуры и перечисления, потому что их легче понимать. Также не забывайте про классы. На практике - большинство пользовательских типов данных, с которыми вы работаете - это структуры и перечисления. Для более детального сравнения см. раздел «Выбор между структурами и классами».

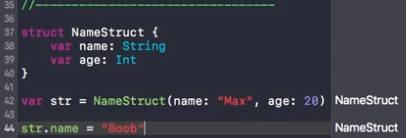


## Поэлементные инициализаторы структурных типов

Все структуры имеют автоматически генерированный "поэлементный инициализатор", который вы можете использовать для инициализации свойств новых экземпляров структуры. Начальные значения для свойств нового экземпляра могут быть переданы поэлементному инициализатору по имени:

let vga = Resolution(width: 640, height: 480)

В отличии от структур, классы не получили поэлементного инициализатора исходных значений. Инициализаторы более подробно описаны в [Инициализация](https://swiftbook.ru/doc/initialization).



HackerRank.com

**Day 0**

## let inputString = readLine()! // get a line of input from stdin and save it to our variable

## print("Hello, World.")

## print (inputString)

**Day 1**

## var i = 4

## var d = 4.0

## var s = "HackerRank "

## // Declare second integer, double, and String variables.

## let i2 = Int (readLine()!)

## let d2 = Double (readLine()!)

## let s2 = readLine()!

## print (i + i2!)

## print (d + d2!)

## print (s + s2)

Альтернатива

## var i = 4

## var d = 4.0

## var s = "HackerRank "

## // Declare second integer, double, and String variables.

## let i2 = readLine()!

## let d2 = readLine()!

## let s2 = readLine()!

## print (i + Int (i2)!)

## print (d + Double (d2)!)

## print (s + s2)

**Day 2**

import Foundation

// Complete the solve function below.

func solve(meal\_cost: Double, tip\_percent: Int, tax\_percent: Int) -> Void {

var mealCost = meal\_cost

var tip = (mealCost \* Double(tip\_percent) / 100)

var tax = (mealCost \* Double(tax\_percent) / 100)

var totalCost = Int(round(mealCost + tip + tax))

print(Int(round(mealCost + tip + tax) ))

}

guard let meal\_cost = Double((readLine()?.trimmingCharacters(in: .whitespacesAndNewlines))!)

else { fatalError("Bad input") }

guard let tip\_percent = Int((readLine()?.trimmingCharacters(in: .whitespacesAndNewlines))!)

else { fatalError("Bad input") }

guard let tax\_percent = Int((readLine()?.trimmingCharacters(in: .whitespacesAndNewlines))!)

else { fatalError("Bad input") }

solve(meal\_cost: meal\_cost, tip\_percent: tip\_percent, tax\_percent: tax\_percent)

**Day 3**

**import** Foundation

**guard** **let** N = Int((readLine()?.trimmingCharacters(**in**: .whitespacesAndNewlines))!)

**else** { fatalError("Bad input") }

**if** N % 2 == 0 {

**if** N <= 5 || N > 20 {

print("Not Weird")

}

**else** {

print("Weird")

}

}

**else** {

print("Weird")

}

**Day 4**

**mport** Foundation

**class** Person {

**var** age: Int = 0

**init**(initialAge: Int) {

**if** initialAge < 0 {

age = 0

print ("Age is not valid, setting age to 0.")

}

**else**{

age = initialAge

}

*// Add some more code to run some checks on initialAge*

}

**func** amIOld() {

**if** age < 13 {

print ("You are young.")

}

**else** **if** (age >= 13) && (age < 18){

print ("You are a teenager.")

}

**else** {

print ("You are old.")

}

*// Do some computations in here and print out the correct statement to the console*

}

**func** yearPasses() {

age += 1

*// Increment the age of the person in here*

}

}

**let** t = Int(readLine()!)!

**for** \_ **in** 0..<t {

**let** age = Int(readLine()!)!

**let** p = Person(initialAge: age)

p.amIOld()

**for** i **in** 1...3 {

p.yearPasses()

}

p.amIOld()

print("")

}

**Day 5**

**import** Foundation

**guard** **let** n = Int((readLine()?.trimmingCharacters(**in**: .whitespacesAndNewlines))!)

**else** { fatalError("Bad input") }

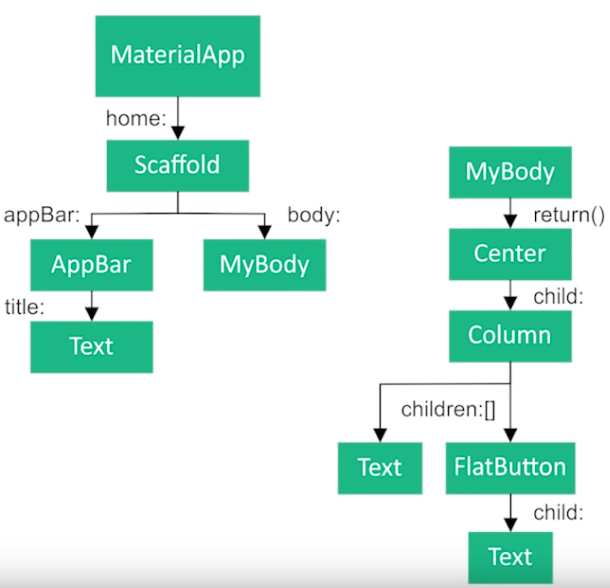
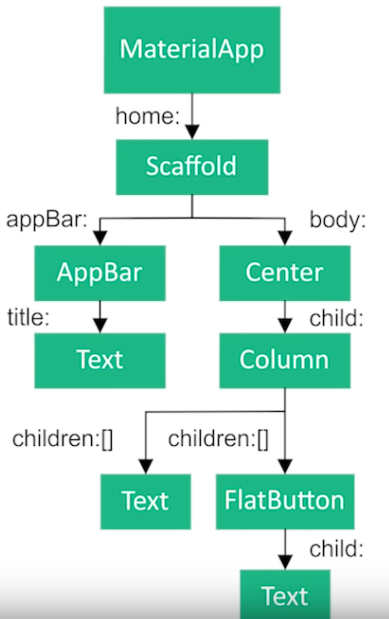
**var** i:Int

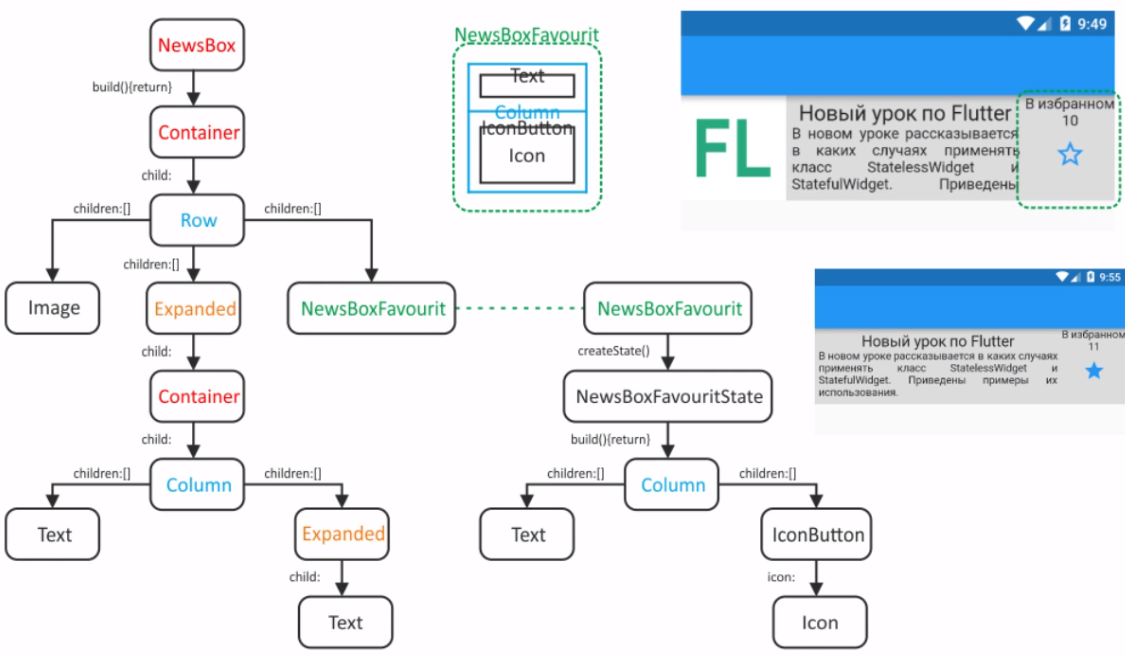
**for** i **in** 1...10 {

print("\(n) x \(i) = \(n \* i)")

}

**Dart**

****

****

var fourDoubles = [Double](repeating: 0.0, count: 4)

print(fourDoubles)

var shoppingList = ["Bread", "Milk"]

shoppingList[0] = "Two apples"

print(shoppingList)

var shoppingList = ["Bread", "Milk"]

shoppingList.append("Flour")

print(shoppingList)

var shoppingList = ["Syrup", "Bread", "Milk", "Apples", "Soup"]

print("Before: \(shoppingList)")

shoppingList[1...2] = ["Bananas", "Oranges"]

print("After: \(shoppingList)")

let r = shoppingList.remove(at:0)

print("After: \(shoppingList)")

// print(r)

var shoppingList = ["Syrup", "Bread", "Milk", "Apples", "Soup"]

let r = shoppingList.removeLast()

shoppingList.insert("New item",at:0)

print(shoppingList.count)

var shoppingList = ["Bread", "Milk", "Syrup", "Apples"]

for item in shoppingList {

print(item)

}

var shoppingList = ["Bread", "Milk", "Syrup", "Apples"]

for (index, value) in shoppingList.enumerated() {

print("Item \(index + 1): \(value)")

// }

for i in 1...10 where i % 2 == 0 {

print("Остаток \(i)")

}

print()

var x = 26, y = 9, z = 8

print("Остаток \(x % y)+\(x % z)")

print()

print("Остаток \(26 % 9)")

var airports = ["TOR": "Toronto", "NY": "New York"]

if let removedValue = airports.removeValue(forKey: "NY") {

print("The removed airport's name is \(removedValue).")

} else {

print("The airports dictionary does not contain a value for NY.")

}

var airports = ["TOR": "Toronto", "NY": "New York"]

for (airportCode, airportName) in airports {

print("\(airportCode): \(airportName)")

}

var airports = ["TOR": "Toronto", "NY": "New York"]

for airportCode in airports.keys {

print("Airport code: \(airportCode)")

}

for airportName in airports.values {

print("Airport name: \(airportName)")

}

var result = 0

for i in 0...4 {

if i == 3 {

result += 10

} else {

result += i

}

print(result)

}

var number = 10

print("number is = \(number)")

var width = 1920

var height = 1080

var numberOfpixel = width \* height

print(numberOfpixel)

Площадь фигуры

var width = 10

var height = 5

var x = 10

var y = 5

var perimeter = 2 \* (width + height)

var bigArea = width \* height

var smallArea = (width - x) \* (height - y)

var area = bigArea - smallArea

print(perimeter)

print(area)

Последняя цифра

var a = 119

print (a % 10 )

Вывод по буквам не работает

for c in "Hello World".characters {

print (c)

}

var peremString = ""

if peremString.isEmpty {

print("is Empty")

}

var peremString = "Hello"

permString.characters.count

peremString.hasPrefix("H")

main() {

print('a single quoted string');

print("a double quoted string");

// Strings can be combined with the + operator.

print("cat" + "dog");

// Triple quotes define a multi-line string.

print('''triple quoted strings

are for multiple lines''');

// Dart supports string interpolation.

var pi = 3.14;

print('pi is $pi');

print('tau is ${2 \* pi}');

var h = """

"Hello"

hgjhk

gkjhg

kkkjn""";

var w = "World";

print('$h $w');

var o = new Object();

print(o.toString());

}

Sunflower 🌻

// Copyright 2019 the Dart project authors. All rights reserved.

// Use of this source code is governed by a BSD-style license

// that can be found in the LICENSE file.

import 'package:flutter/material.dart';

import 'dart:math' as math;

final Color primaryColor = Colors.red;

final TargetPlatform platform = TargetPlatform.android;

void main() {

runApp(Sunflower());

}

class SunflowerPainter extends CustomPainter {

static const seedRadius = 3.0;

static const scaleFactor = 7;

static const tau = math.pi \* 2;

static final phi = (math.sqrt(5) + 1) / 2;

final int seeds;

SunflowerPainter(this.seeds);

@override

void paint(Canvas canvas, Size size) {

var center = size.width / 2;

for (var i = 0; i < seeds; i += 3 ) {

var theta = i \* tau / phi;

var r = math.sqrt(i) \* scaleFactor;

var x = center + r \* math.cos(theta);

var y = center - r \* math.sin(theta);

var offset = Offset(x, y);

if (!size.contains(offset)) {

continue;

}

drawSeed(canvas, x, y);

}

}

@override

bool shouldRepaint(SunflowerPainter oldDelegate) {

return oldDelegate.seeds != this.seeds;

}

// Draw a small circle representing a seed centered at (x,y).

void drawSeed(Canvas canvas, num x, num y) {

var paint = Paint()

..strokeWidth = 2

..style = PaintingStyle.fill

..color = primaryColor;

canvas.drawCircle(Offset(x, y), seedRadius, paint);

}

}

class Sunflower extends StatefulWidget {

@override

State<StatefulWidget> createState() {

return \_SunflowerState();

}

}

class \_SunflowerState extends State<Sunflower> {

double seeds = 100.0;

int get seedCount => seeds.floor();

@override

Widget build(BuildContext context) {

return MaterialApp(

debugShowCheckedModeBanner: false,

theme: ThemeData().copyWith(

platform: platform,

brightness: Brightness.dark,

sliderTheme: SliderThemeData.fromPrimaryColors(

primaryColor: primaryColor,

primaryColorLight: primaryColor,

primaryColorDark: primaryColor,

valueIndicatorTextStyle: DefaultTextStyle.fallback().style,

),

),

home: Scaffold(

appBar: AppBar(title: Text("Sunflower")),

drawer: Drawer(

child: ListView(

children: [

DrawerHeader(

child: Center(

child: Container(

child: Text(

"Sunflower 🌻",

style: TextStyle(fontSize: 32),

),

),

),

),

],

)),

body: Container(

constraints: BoxConstraints.expand(),

decoration:

BoxDecoration(border: Border.all(color: Colors.transparent)),

child: Column(

crossAxisAlignment: CrossAxisAlignment.center,

mainAxisAlignment: MainAxisAlignment.start,

children: [

Container(

decoration: BoxDecoration(

border: Border.all(color: Colors.transparent)),

child: SizedBox(

width: 700,

height: 700,

child: CustomPaint(

painter: SunflowerPainter(seedCount),

),

),

),

Text("Showing $seedCount seeds"),

ConstrainedBox(

constraints: BoxConstraints.tightFor(width: 400),

child: Slider.adaptive(

min: 0,

max: 396,

value: seeds,

onChanged: (newValue) {

setState(() {

seeds = newValue;

});

},

),

),

],

),

),

),

);

}

}

Sunflower 🌻

// Copyright 2019 the Dart project authors. All rights reserved.

// Use of this source code is governed by a BSD-style license

// that can be found in the LICENSE file.

import 'package:flutter/material.dart';

import 'dart:math' as math;

final Color primaryColor = Colors.green;

final TargetPlatform platform = TargetPlatform.android;

void main() {

runApp(Sunflower());

}

class SunflowerPainter extends CustomPainter {

static const seedRadius = 3.0;

static const scaleFactor = 2;

static const tau = math.pi \* 2;

static final phi = (math.sqrt(5) + 1) / 3;

final int seeds;

SunflowerPainter(this.seeds);

@override

void paint(Canvas canvas, Size size) {

var center = size.width / 2;

for (var i = 0; i < seeds; i += 3 ) {

var theta = i \* tau / phi;

var r = math.sqrt(i) \* scaleFactor;

var x = center + r \* math.cos(theta);

var y = center - r \* math.sin(theta);

var offset = Offset(x, y);

if (!size.contains(offset)) {

continue;

}

drawSeed(canvas, x, y);

}

}

@override

bool shouldRepaint(SunflowerPainter oldDelegate) {

return oldDelegate.seeds != this.seeds;

}

// Draw a small circle representing a seed centered at (x,y).

void drawSeed(Canvas canvas, num x, num y) {

var paint = Paint()

..strokeWidth = 2

..style = PaintingStyle.fill

..color = primaryColor;

canvas.drawCircle(Offset(x, y), seedRadius, paint);

}

}

class Sunflower extends StatefulWidget {

@override

State<StatefulWidget> createState() {

return \_SunflowerState();

}

}

class \_SunflowerState extends State<Sunflower> {

double seeds = 1.0;

int get seedCount => seeds.floor();

@override

Widget build(BuildContext context) {

return MaterialApp(

debugShowCheckedModeBanner: false,

theme: ThemeData().copyWith(

platform: platform,

brightness: Brightness.dark,

sliderTheme: SliderThemeData.fromPrimaryColors(

primaryColor: primaryColor,

primaryColorLight: primaryColor,

primaryColorDark: primaryColor,

valueIndicatorTextStyle: DefaultTextStyle.fallback().style,

),

),

home: Scaffold(

appBar: AppBar(title: Text("Sunflower")),

drawer: Drawer(

child: ListView(

children: [

DrawerHeader(

child: Center(

child: Container(

child: Text(

"Sunflower 🌻",

style: TextStyle(fontSize: 64),

),

),

),

),

],

)),

body: Container(

constraints: BoxConstraints.expand(),

decoration:

BoxDecoration(border: Border.all(color: Colors.transparent)),

child: Column(

crossAxisAlignment: CrossAxisAlignment.center,

mainAxisAlignment: MainAxisAlignment.start,

children: [

Container(

decoration: BoxDecoration(

border: Border.all(color: Colors.transparent)),

child: SizedBox(

width: 700,

height: 700,

child: CustomPaint(

painter: SunflowerPainter(seedCount),

),

),

),

Text("Showing $seedCount seeds"),

ConstrainedBox(

constraints: BoxConstraints.tightFor(width: 200),

child: Slider.adaptive(

min: 0,

max: 12500,

value: seeds,

onChanged: (newValue) {

setState(() {

seeds = newValue;

});

},

),

),

],

),

),

),

);

}

}